Technische Fachhochschule Berlin
Fachbereich 13

ENTWICKLUNG EINES KIOSK-SYSTEMS UNTER WINDOWS

Diplomarbeit
von Percy Wippler
Matrikelnummer 521615

Betreuung durch Prof. Dr. K. H. Sturm

Wintersemester 1994/95

Ich bestätige, daß ich diese Arbeit selbständig und nur mit den angegebenen Quellen angefertigt habe!

Vorwort

“Ich hätte noch eine ganz andere Diplomarbeit für Sie!” lautete ungefähr der Satz meines betreuenden Professors, der diese Arbeit zur Folge haben sollte. Die Idee ein Kiosk-System zu entwickeln hörte sich durchaus interessant an, auch wenn ich damals noch nicht wußte, was ein Kiosk-System überhaupt ist. Die kurze Einführung in Touchscreens, Multimedia und graphische Benutzerführung erweckte in mir großes Interesse an dieser Arbeit. Während meiner gesamte Computerzeit (angefangen mit einem Commodore C16) lag meine Hauptneigung in der Grafikprogrammierung. Der Anreiz, daß das Produkt eventuell von der Firma Vobis genutzt werden würde, ließ mich sehr schnell zu dem Entschluß kommen, ein Kiosk-System zu entwickeln.
Dank sagen möchte ich Herrn Prof. Dr. Sturm für seine sehr intensive Betreuung und Motivation, die nicht aus Vorsagen, sondern in erster Linie aus eindringlichen Anregungen bestand. Durch die freie Hand, daß gesamte Projekt vom Gehäuse bis zur Software zu planen und zu organisieren, war der Lerneffekt bezüglich Projekt-Management und Software-Engineering sehr hoch.
Die nachfolgende Dokumentation meiner Arbeit beschränkt sich jedoch auf die Entwicklung der Software und deren Hardwarevoraussetzungen, da dies den Hauptteil der Arbeit ausmachte und es sich in meinem Studiengang um technische Informatik handelt.



Inhalts-verzeichnis

Inhaltsverzeichnis



Einleitung

Einleitung

Multimedia

Multimedia! Heute ein sehr populäres Modewort, wird im Zusammenhang einer immer mehr in den Vordergrund dringenden Computertechnik verwendet.

Multi [lat.] steht für mehrere oder viel[1]
Media [lat.] ist die Mehrzahl eines Mediums [lat. das in der Mitte Befindliche] *bildungssprachlich:* vermittelndes Element, Mittel zur Weitergabe oder Verbreitung von Information durch Sprache, Gestik, Mimik, Schrift, Bild, Musik.[1] *Meyers Lexikon*

Da erscheint die Wortkombination von Multi und Media eigentlich überflüssig, weil Media schon mehrere beinhaltet. Und doch ist dies nicht ganz sinnlos, da über Computer, auch im Heimbereich, diese Medien gleichzeitig über einen Bildschirm und Lautsprecher ausgegeben werden können. Das Fernsehen benutzt in diesem Sinne schon seit je her diese Technik mit dem Nachteil, daß der Informationsempfänger keine Eingriffsmöglichkeit in das Präsentierte hat. Nachrichten müssen in der Reihenfolge und dem Tempo aufgenommen werden, wie es die Redaktion vorsieht.
In einem **Kiosk-System** sieht dies anders aus. Der Vorteil eines solchen Systems liegt in der Präsentationsmöglichkeit von **Multimedia** und der Möglichkeit einer **Interaktion** (Wechselbeziehungen, besonders die Kommunikation zwischen Individuen innerhalb einer Gruppe[1]). Die Kommunikation findet zwischen einer Person und dem Computer über Eingabeperipherien statt, die in verschiedensten Formen vorliegen können. Für das entwickelte Kiosk-System, das den Namen **KIOSK V1.0** erhalten hat, wird die Interaktion über einen **Touchscreen** gewährleistet. Die Person betätigt auf dem Bildschirm sichtbare Felder, in der Hoffnung die von ihm gewünschte Information zu erhalten. Der Vorteil dieser Technik liegt in der direkten Funktionsvisualisierung eines Tasters (Buttons).
Die Einsatzmöglichkeiten der Multimedia-Applikationen sind vielfältig, vor allem in der Arbeitswelt. Firmeninterne Kommunikation, Ergebnispräsentation, interaktive Nachschlagewerke, Werbung, Kataloge, Produktdemonstrationen, Ausbildungswesen etc.[2]
.

[1] s.a. PC-Bibliothek Meyers Lexikon, 1993
[2] vgl. Faszination Multimedia, 1993, S. 8-9

betriebliche Nutzung

Die geschäftliche Nutzung eines Kiosk-System in einem Dienstleistungs- und Handelsbetrieb entlastet den Berater oder Verkäufer bei der Kundenabfertigung. Die Verfassung und die Art der Fragestellung des Kunden hat keinen Einfluß auf die Kommunikation. Das Kompensieren persönlicher Berührungsängste erleichtern dem Kunden den Erhalt von Informationen.
Die Wirkungsweise von Multimedia und Interaktion steht dem handelsüblichen Informationsfluß in folgender Weise gegenüber:

Lesen: 10%
Hören 20%
Sehen 30%
Hören und Sehen 50%
Selber sagen 70%
Selber tun 90%



Anforderungsdefinition und Abgrenzung

1.1 Anforderungsdefinition

Es ist ein Kiosk-System zu entwickeln, welches unter MS-Windows 3.1 auf einem IBM-kompatiblen AT unter Verwendung eines Touchscreens arbeitet. Es soll keine weitere Eingabe-Peripherie verwendet werden. Zur Ausgabe dienen der Monitor, Audio, und optional ein Drucker, ggf. auch ein Diskettenlaufwerk. Das System soll sich den vom Anbieter bereit gestellten Daten anpassen. Die unterschiedlichen Themen- und Informationsbereiche werden vom Anbieter in dafür vorgesehenen Verzeichnissen gespeichert. Das System soll über Touchbuttons, durch Auswahl des Kunden, die individuellen Informationen anzeigen. Der Abbruch des Programms und eine Umkonfigurierung ist nur dem Anbieter vorbehalten und durch ein Paßwort geschützt. Über einen spezifischen Treiber sollen geschäftsbedingte Informationen wie Lagerbestand und Artikelpreis von anderen Systemen integrierbar sein. Das System ist in einem abschließbaren Kiosk-Terminal-Gehäuse zu integrieren und soll im Stehen zu bedienen sein. Das Programm soll die Eingabe eines stichworthaften Wunschkataloges der Kunden erlauben, was der jeweiligen Filiale für Artikel fehlen.

1.2 Abgrenzung

Die Eingabe, Gestaltung und Aufarbeitung der wiederzugebenden Informationen sind mit Ausnahme von Testdateien nicht Inhalt dieser Arbeit. Weitere integrierbare Elemente, wie Netzwerk-Kommunikation zwischen verschiedenen Filialen, Kundenspezifizierung durch Magnetstreifenleser, Bestellautomation, Sprachsteuerung, Tools zur Datenaufbereitung u.ä. sind ebenfalls in dieser Ausführung nicht berücksichtigt.



Lösungs-
Ansätze

2.1 Hardwarevoraussetzung

Mit der MPC-Norm (*Multimedia Personal Computer*), die auf eine Initiative von Microsoft zurückgeht, wurde 1993 im Level II eine minimale Konfiguration definiert, die ein Computersystem benötigt, um sich Multimedia nennen zu dürfen. Diese besteht aus[1]:

Die MPC-Norm

- Computer IBM PC oder kompatibel
- 80486SX / 25 MHz - Prozessor
- VGA-Grafikkarte und passendem Monitor für 65.535 (64K) Farben
- 4 MByte RAM
- 160 MByte HDD
- 2,5” Laufwerk - 1,44 MByte Kapazität
- Maus mit zwei Tasten
- Eine serielle und eine parallele Schnittstelle
- 16-Bit-Soundkarte
- CD-ROM-Laufwerk mit 300 KByte/s Datentransfer und einer durchschnittlichen Zugriffszeit von maximal 400 ms, bei Auslastung der CPU von weniger als 60%.
- Windows 3.1 oder neuer

Da diese Minimalausstattung für den hohen Datentransfer bei Bildern und Videos nicht die mögliche Geschwindigkeit liefert, wie es bei anderen Computern der Fall ist, wird eine Konfiguration mit 80486-DX2-66-Mhz-Prozessor und 16 MByte RAM gewählt. Die Festplatte soll so groß sein, daß eine komplette CD überspielt werden kann. Die Wahl trifft auf ein 1-GByte-HDD. Das CD-ROM wird für einen schnellen Transfer als Doublespeed-Laufwerk gewählt.
Das technische Verfahren des Touchscreens beruht auf kapazitiver Positionsermittlung, da es als am wenigsten anfällig gilt. Die Datenübertragung erfolgt über die serielle Schnittstelle. Jede Berührung simuliert über einen speziellen Treiber die Mausfunktionen als Drag-And-Drop.

[1] Faszination Multimedia, 1993, S. 10-11

2.2 Entwicklungstools

Zur Realisierung des Systems können verschiedene Werkzeuge zum Einsatz kommen. Zum einen über die Benutzung von **CAP**-Programmen (*Computer Aided Presentation*). CAP entwickelte sich im Laufe der letzten Jahre aus dem, was man einfach Präsentationssoftware oder *Shows* nannte. Ihre Rolle besteht darin, verschiedene Bildschirmansichten oder verschiedene, vorher erstellte Bilder aneinander zu reihen, wobei Übergänge mit bestimmten Spezialeffekten ermöglicht werden. Der Weg von CAP-Programmen hin zu Multimedia-Applikationen wird zwar angegangen, ist aber derzeit immer noch eine Tendenz. Der zweite Nachteil ist die Integration der Daten vor einer Präsentation. Somit läßt sich eine individuelle Anpassung nur mühsam und bei Änderung der gesamten Präsentation realisieren.

Als zweite Möglichkeit steht dem die Entwicklung einer neuen Software mittels **OpenFace (Toolbook), Visual Basic, Borland (Turbo) Pascal, Quick C und Visual C** gegenüber. Diese Programmiersprachen können über das **MCI** (*Multi Control Interface*) Multimediadaten darstellen. **Toolbook** ist zur Erstellung von Informationsseiten gut geeignet. Zwischen diesen kann einfach gewechselt werden. Das Handling der Windows-Botschaften erfolgt auf komfortable Weise. Jedoch ist dieses Werkzeug als **Interpretersprache** für die umfangreiche Anwendung der Multimedia zu langsam. Wesentlich bessere Ergebnisse liefern **Borland Pascal** und die **C-Compiler**, jedoch erfordert die Erstellung einer vielseitigen Benutzeroberfläche gute Kenntnisse in **objektorientierter Programmierung (OOP)**, die nicht Inhalt des technischen-Informatik-Studiums war. Die beste Lösung zur Erstellung einer Benutzeroberfläche unter Windows liefert derzeit **Visual Basic 3.0**. Hier wird durch erweiterbare Werkzeuge und einer unmittelbaren (visuellen) Gestaltungsmöglichkeit vielzähliger Objekte die Arbeit erheblich erleichtert, da weder Programmcode noch eine Compilierung notwendig ist. Nachteil dieser Programmiersprache ist die zurückstehende Rechengeschwindigkeit. Somit ergibt sich eine optimale Lösung aus einer **Kombination** von **Basic** und **Pascal** bzw. **C**. Hier wurde für Borland Pascal 7.0 entschieden, weil bei

mir die Kenntnisse weiter als in C gehen, seine Struktur übersichtlicher ist und eine Compilierung schneller abläuft.
In C gäbe es die Möglichkeit neue Zusatz-Steuerelemente (**VBX**) zu erzeugen, die in Visual Basic leichter als andere **DLL** zu integrieren sind. Dies macht aber einen erheblichen Aufwandunterschied aus und lohnt sich nur dann, wenn erstellte VBX-Dateien in weiteren Software-Projekten zur Wirkung kommen. Die Kommunikation zwischen Basic und Pascal erfolgt über in Pascal zu erstellende Dynamic Link Librarys (DLL).

2.3 Methodische Ansätze

Zur Modellierung der Software gibt es verschiedene Möglichkeiten, von den strukturierte Analyse (**SA**) mit Datenflußdiagrammen (**DFD**), Zustands-Aktions-Modelle (**ZA**) bzw. Interaktionsdiagramme (**JAD)**, **Petri-Netze** und **SADT** zur Diskussion stehen.
Abhängig von Windows-Spezifikationen sind dem System Ereignissteuerungen zu Grunde gelegt. D.h. jedes Objekt (Fenster, Button, Bild, Animation ...) hat immer einen **Zustand**, der durch Eintreten eines **Ereignisses** (Button wurde betätigt, Bild wurde dargestellt, Animation wurde beendet ...) ggf. in einen anderen wechselt. Somit erfolgt eine Steuerung des Systems nicht offensichtlich durch Datentransfer (dies ist in Windows integriert, und muß nicht neu definiert werden). Zustandsänderungen erfolgen wie oben beschrieben als Reaktion auf Ereignisse.
SA beschreibt ein System in erster Linie durch Datenflüsse, Datenspeicher und deren Knotenpunkte. Im Kiosk-System erfolgt der Hauptteil des Datentransfers über Windows- und DOS-Funktionen (Hierarchie über Verzeichnisse, Makroabarbeitung über Windows-INI-Dateien). Es kommt lediglich beim Eintreten eines Ereignisses u.U. zur **Aufbereitung von Daten** und somit wieder zu Zustandsänderungen.
SADT beschreibt eine Prozessorganisation in Abhängigkeit von Steuerungen, Mechanismen und Dateninhalten. Grundsätzlich wirkt auf ein Objekt zu einem Zeitpunkt immer nur ein Ereignis ein. Es ergibt sich

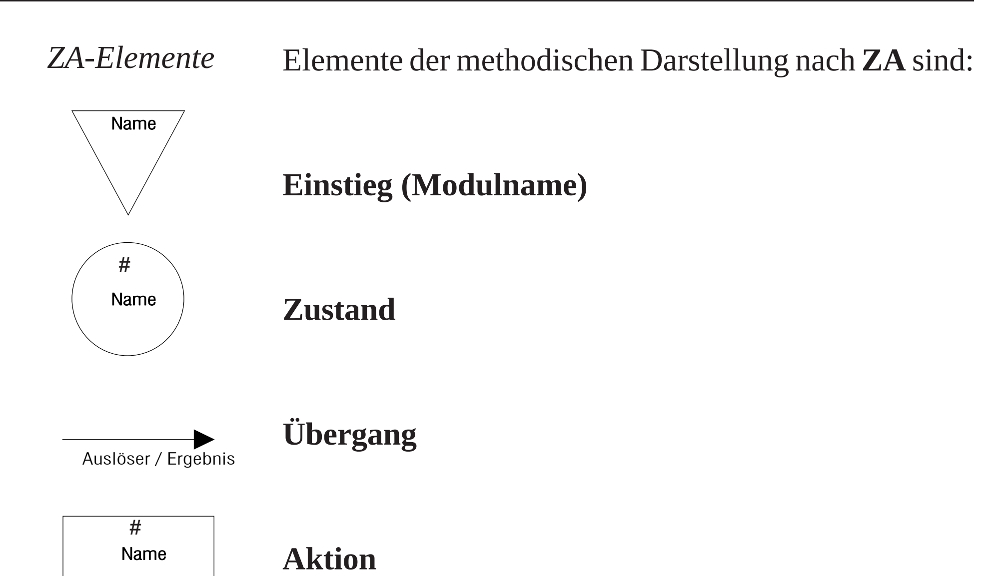
zwar eine mögliche Abhängigkeit vom Datenzustand, aber mehrere Mechanismen und Steuerungen, wie es unter SADT beschreibbar ist, finden nicht gleichzeitig statt.

Zustands-Aktions-Modelle werden den Anforderungen, sofern keine Parallelprozesse (Meta-Zustände) modelliert werden müssen, gerecht. Im System kommt es zwar zur Abarbeitung von multi-tasking-ähnlichen Prozessen, jedoch übernimmt deren Kontrolle Windows selbst. Resultate der Animations- und Klangverwaltung werden von Windows als Ereignis gemeldet (Bsp. Klang beendet). Auf die umfangreichere Modellierung mittels **Petri-Netze** kann verzichtet werden, da die einfachere Form dieser, als **ZA**, den Bedingungen genügt und Petri-Netze während des Studiums nicht ausreichend vertieft wurden.

2.3.1 Elemente des ZA

ZA-Elemente

Elemente der methodischen Darstellung nach **ZA** sind:



Einstieg (Modulname)

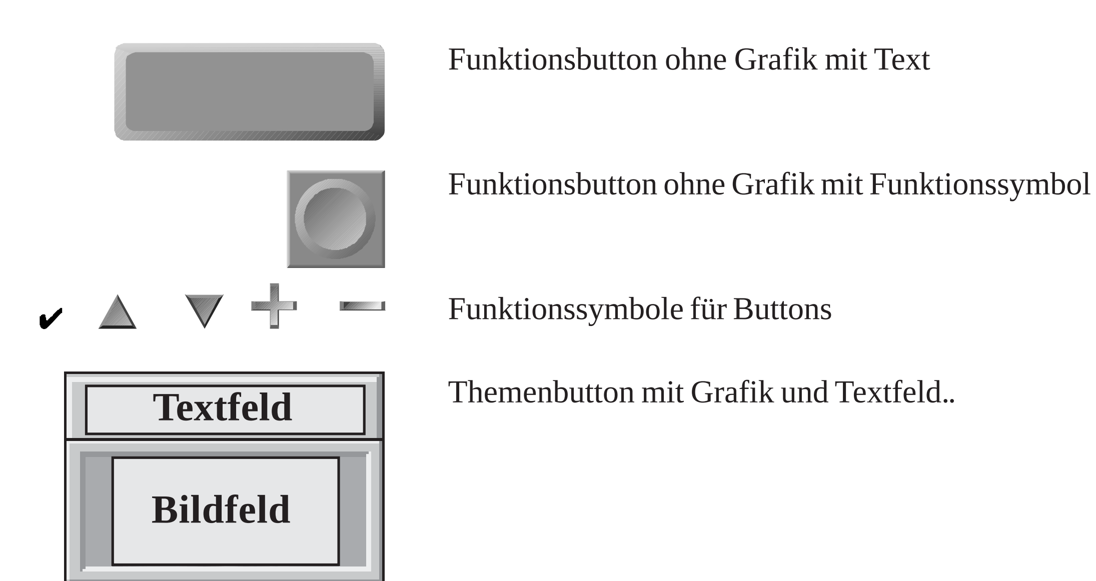
Zustand

Übergang

Aktion

2.4 Ansatz zur Benutzeroberfläche

Die Bedienung des Systems soll möglichst einfach sein, und nur geringe Anforderung bezüglich des Vorwissens durch den Kunden haben. Eine einfache Bedienung beschränkt sich auf eine Eingabeeinheit. Kann diese flexibel gestaltet werden (Änderung des Aussehens oder der Beschreibung) ist eine eindeutige Funktionszuordnung gegeben. Daher ist eine Betätigung von grafischen Elementen mittels **Touchscreen** der Eingabe durch eine **Tastatur** vorzuziehen. Hierbei müßte der Benutzer erst die Wirkung einer bestimmten Taste nachlesen und erlernen. Schwieriger und ungewohnter wäre eine Eingabe über **Maus, Trackball oder Joystick**. Diese Art der Eingabe erfordert eine längere Übungszeit, was nicht Sinn eines zeitsparenden Informationssystem ist.
Sinnvoll ist eine Anpassung der Grafiken an reelle alltägliche Elemente. Somit sollten Schalter als solche eindeutig identifizierbar sein, so daß der Benutzer nicht lange raten muß, wo er das System zu bedienen hat. Ebenfalls muß jedem Objekt beim flüchtigen Hinsehen eine eindeutige Funktionalität zu entnehmen sein. Ein Feld zur Textein- und ausgabe sollte nicht wie eine Schalt- oder Malfläche aussehen. Eine Beschreibung des Ergebnisses bei einer Betätigung muß kurz und eindeutig sein (Stichwort). Eine Unterstützung von **Grafik innerhalb einer Schaltfläche** erleichtert die Funktionserkennung durch den Benutzer.



Funktionsbutton ohne Grafik mit Text

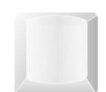
Funktionsbutton ohne Grafik mit Funktionssymbol

Funktionssymbole für Buttons

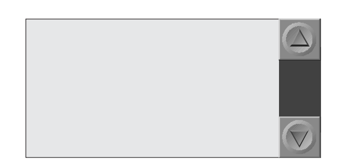
Themenbutton mit Grafik und Textfeld..



Scroll-Leiste mit 2 Funktionsbutton



Buchstabenbutton für Tastatur



Text- und Listenfeld mit Scroll-Leiste

Infoleiste mit einer zentrierten Textzeile.

Schaltflächen sind so anzuordnen, daß bei einer Betätigung der Arm nicht den gesamten Bildschirm überdeckt. Somit ist ebenso eine Funktion zu überdenken, die eine Umgestaltung des Bildschirms zur Folge hat. Für Rechtshänder auf der rechten Seite des Bildschirms Funktionstasten und links die Ausgabe, für Linkshänder entgegengesetzt.
Das System soll sich dem Benutzer anpassen, nicht der Benutzer dem System.





Buttons für die Rechts- Linksanordnung

2.5 Benutzerschnittstellen

Die Einbindung von Medien soll im System flexibel gestaltbar sein. Eine aufwendige Programmierung soll umgangen werden, aber eine Steuerung über Makros möglich sein. Zwei Möglichkeiten stehen zur Diskussion. Erstens kann ein Makro sequentiell ausgelesen und abgearbeitet werden. Hierbei kann eine beliebige Folge von Bildern, Animationen, Klängen, Texten, Pausen, und Wiederholungen ausgegeben werden. Zum zweiten steht ein hierarchiefreies Abarbeiten eines Makros zur Auswahl. Dieses Prinzip läßt jede Zuweisung und somit jede Art von Medium pro Thema genau einmal zu, was der Nachteil daran ist. Der Vorteil ist die einfache technische Umsetzung für Entwickler und Anwender, da sich diese Makros nach dem Prinzip der Windows-INI-Dateien erstellen und abfragen lassen und dies somit keine Fehleranfälligkeit zur Folge hat. Bei der Planung wurde für die zweite Variante entschieden.

Da das Kiosk-System mit anderen Systemen kommunizieren können soll, aber nicht bekannt ist, wie diese Kommunikation aussieht, ist eine Schnittstelle zu definieren, die über die Entwicklung eines speziellen Treibers erfolgen kann. Da in Windows alle Programmteile aus Dynamic Link Librarys bestehen (DLL, DRV, EXE, VBX ...) liegt die Nutzung dieses Prinzips nahe. In einer DLL können beliebige Funktionen, die auch Hardware-Steuerungen übernehmen können, integriert sein.



Entwurf

3.1 Benutzerschnittstellen

Die Schnittstellen für den Benutzer sind in zwei Bereiche zu gliedern:

1. Schnittstelle: Makros
2. Schnittstelle: Benutzeroberflächen

3.1.1 Makros

INFO.INI

Makros der entwickelten Software sind im ANSI-Format gespeicherte Textdateien, die den Namen **INFO.INI** bzw. **INDEX.INI** erhalten. Eine INFO.INI weist dem Kiosk-System Aktionen zu, die einer Buttonbetätigung oder einem Verzeichniswechsel folgen sollen. Aktionen werden je nach Ursache über **[Action]** bzw. **[Button*Name*]** eingeleitet. **[Action]** ist die Reaktion auf einen Verzeichniswechsel, **[Button*Name*]** beschreibt die Optionen einer Schaltfläche und die Reaktionen auf eine Betätigung dieser. ***Name*** entspricht einer beliebigen aber eindeutigen Zuordnung. Jeder vergebene Name darf pro INFO.INI genau ein Mal vorkommen. Über ***Name*** werden die Reaktionen auf einen Themenbutton-Mausklick im Makro lokalisiert.
Den in eckigen Klammern stehenden Einleitungen folgen Steuerbefehle und Zuweisungen von auszugebenden Medien. Diese können sowohl bei **[Action]** als auch bei **[Button*Name*]** sein:

Makrobefehle (1. Teil)

picture=*Pfad + Bilddatei*
animation=*Pfad + Videodatei*
animreplay=*Anzahl*
text=*Pfad + Textdatei*
sound=*Pfad + Klangdatei*
soundreplay=*Anzahl*
sounddelay=*Sekunden*
Sound_While_Anim=*Option*
Sound_After_Anim=*Option*
Picture_While_Sound=*Option*
print=*Option*

Kursiv dargestellte Abschnitte entsprechen variablen Angaben. Normal dargestellte stehen in ihrer Zeichenfolge fest. Die Groß-/Kleinschreibung kann variieren. Alle Angaben für ***Pfad*** genügen den DOS-Konventionen.

Videodatei-Format

Animationen müssen als Video-for-Windows-Datei (AVI) vorliegen. Die Abspielorganisation wird vom Windows-MCI-Treiber vorgenommen. Eine Animation wird nur dann abgespielt, wenn ihre Größe den zur Ausgabe vorgesehen Platz nicht überschreitet. Dieser umfaßt maximal 5/8 der Bildschirmbreite und 1/2 der Bildschirmhöhe. Sind alle Wiederholungen einer Animation abgespielt, wird diese vom Bildschirm und aus dem Speicher entfernt.

Animationsoption

Eine Animation wird so oft abgespielt, wie es der Option **animreplay** zugewiesen ist. Eine negative Zahl läßt die Animation endlos wiederholen. Wird die Zuweisung weggelassen, spielt die Animation genau ein Mal.

Bilddatei-Format

Eine Bilddatei kann als Windows-Bitmap oder Windows-Metafile (inkl. Header) vorliegen. Ausmaße und Farbtiefe sind beliebig. Bilder werden maximal mit den Ausmaßen 5/8 der Bildschirmbreite und 1/2 der Bildschirmhöhe dargestellt. Die Farben werden der Windows-Einstellung angepaßt. Bilder werden erst dann sichtbar, wenn keine Animation zugewiesen oder eine Animation vom Bildschirm entfernt wurde.

Klangdatei-Format

Klänge müssen im Windows-Wave-Format vorliegen. Die Abspielorganisation wird vom Windows-MCI-Treiber vorgenommen. Sind alle Wiederholungen eines Klanges abgespielt, wird dieser aus dem Speicher entfernt.

Klangoptionen

Ein Klang wird so oft abgespielt, wie es der Option **soundreplay** zugewiesen ist. Eine negative Zahl läßt den Klang endlos wiederholen. Wird diese Zuweisung weggelassen, spielt der Klang genau ein Mal. Zwischen Klangwiederholungen kann eine in Sekunden anzugebende Pause erfolgen. Diese wird über **sounddelay** gesetzt. Wird diese Zuweisung weggelassen, erfolgt keine Pause zwischen den Wiederholungen.

Ist **Sound_After_Anim** ein Wert ungleich Null zugewiesen, beginnt das Spielen eines Klanges erst nach dem Beenden einer Animation. Wird diese Zuweisung weggelassen, spielt ein Klang direkt nach dem Laden.

Ist **Picture_While_Sound** ein Wert ungleich Null zugewiesen, wird ein Bild mit dem Beenden des Klanges vom Bildschirm genommen.

Für **Sound_While_Anim** können die Werte 0, 1 oder 2 zugewiesen werden. Bei einer Zuweisung von Eins, beginnt ein Klang mit jeder Animationswiederholung von vorne. Die Klangwiederholungen sind durch **soundreplay** begrenzt. Der aktuelle Abspielstatus und eine Verzögerung haben keinen Einfluß.
Eine Zuweisung von Zwei läßt einen Klang nach seinen gesetzten Optionen solange abspielen, bis die Animation beendet wurde oder alle Klangwiederholungen ausgeführt wurden.

Textdatei-Format

Texte können im ANSI-Format als unformatierter Text, bzw. im “MS Rich Text Format” als formatierter Text ausgegeben werden. Für die Textausgabe steht ein Bereich zur Verfügung, der halb so breit wie der Bildschirm ist. Die Höhe des sichtbaren Bereiches ist abhängig von der Höhe eines dargestellten Bildes bzw. einer Animation. Der Textausgabebereich kann maximal die gesamte Bildschirmhöhe abzüglich einer Informationszeile einnehmen. Ist der gesamte Text nicht gleichzeitig sichtbar, läßt sich dieser über eine Scroll-Funktion verschieben.

Bild statt Text

Anstatt einer Textdatei können auch Bilder in diesem Ausgabebereich dargestellt werden. Bilder müssen als BMP- bzw. WMF-Datei (inkl. Header) vorliegen. Die Bildgröße wird bei Erhaltung der Bildproportionen in das Ausgabefenster eingepaßt. Bilder lassen sich ebenfalls über die Scrollfunktion verschieben.
Andere Formate werden als ANSI angesehen und entsprechend ausgegeben.

Alle Medien können gleichzeitig, mit Ausnahme der Darstellung von Bildern und Animationen, wiedergegeben werden.

Druckoption

print=1 gestattet das Ausdrucken der aktuell dargestellten Medien. Ist **print=0** oder wird **print** weggelassen, erhält der Anwender nicht diese Möglichkeit.

Makrobefehle (2. Teil)

Einem Button können außerdem noch folgende Zuweisungen folgen:

title=*beliebiger Text*
thumbnail=*Pfad + Bilddatei*
dir=*Verzeichnisname*

Aussehen eines Buttons

Ein *beliebiger Text,* der **title** zugewiesen ist, wird bei der Erzeugung eines Buttons als Titel einzeilig eingesetzt. Die Schriftattribute des Titels entsprechen den unter KIOSK initialisierten. Paßt ein Titel von der Länge nicht in den Button, wird die Schriftart verkleinert. Paßt der Titel bei kleinstem Schriftmaß nicht, ist der Titel rechts beschnitten.
Ist **thumbnail** eine gültige Bilddatei (BMP oder WMF) zugewiesen, wird diese, bei Erhaltung der Bildproportionen, in die dafür vorgesehene Fläche bei der Erzeugung eingepaßt. Anstelle von Pfad setzt ein **->** (Minus + größer als) das unter **dir** zugewiesene Verzeichnis ein. Die unter **picture** angegebene Bilddatei wird als Buttonbild verwandt, wenn die Zuweisung **thumbnail=->** (ohne Bildnamen) erfolgt.

Verzeichniswechsel

Ist dem Befehl **dir** ein gültiges Verzeichnis zugeordnet, Ausgangspunkt ist das aktuelle Verzeichnis, wird bei einer Buttonbetätigung in dieses gewechselt. Es werden keine zugewiesene Media ausgegeben. Bei jedem Verzeichniswechsel wird die dort befindliche INFO.INI gelesen und umgesetzt.
In der INFO.INI darf jeder in eckigen Klammern stehende Bezeichner nur genau einmal vorkommen. Eine Reaktion auf das Öffnen eines Verzeichnisses ist somit über **[Action]** nur einmal deklarierbar.

INDEX.INI

Über die INDEX.INI wird angegeben, welche Themen einem bestimmten Bereich zugeordnet sind, in welchem Verzeichnis das Thema steht und welcher Button die Themendaten beschreibt.
Der Aufbau der INDEX.INI sieht wie folgt aus:

[*Bereich1*]
Schlüsselwort1=pfad,Buttonname1
Schlüsselwort2=pfad,Buttonname2
Schlüsselwort3=pfad,Buttonname3
⋮
[*Bereich2*]
Schlüsselwort1=pfad,Buttonname1
Schlüsselwort2=pfad,Buttonname2
Schlüsselwort3=pfad,Buttonname3
⋮

Für **[*Bereich#*]** kann ein beliebiger Name angegeben werden, der in einem Bereichslistenfeld ohne eckige Klammern wiedergegeben wird. Bei der Auswahl des

Bereiches werden in einem Themenlistenfeld die ***Schlüsselwörter#*** aufgelistet. Bei der Auswahl eines Schlüsselwortes wird in das unter ***Pfad*** angegebene Verzeichnis gewechselt. Ausgangspunkt für ***Pfad*** ist das Datenverzeichnis. Für weitere Angaben gelten die DOS-Konventionen. ***Pfad*** folgt, durch ein Komma getrennt, der Name des Buttons, der in INFO.INI des gewählten Verzeichnisses hinter der Zuordnung **[Button*Name*]** steht. Die Groß-/Kleinschreibung bleibt unbeachtet.
Entfällt ***Pfad***, wird in das Datenverzeichnis gewechselt. Entfällt die Angabe eines Buttonnames wird nur das angegebene Verzeichnis geöffnet.

3.1.1 Benutzeroberflächen

Die in Kapitel 2 definierten Objekte der grafischen Benutzeroberfläche sind immer Teil eines Fensters. Fenster können sich überlagern. Ein Fenster, welches durch Überlagerung in den Hintergrund gerät, wird automatisch inaktiv und hält diesen Zustand, bis das überlagernde Fenster geschlossen wird. Jedes Fenster enthält mindestens ein Objekt zum Schließen desselben. Alle Fenster sind nicht in der Größe und Position durch den Benutzer veränderbar. Sie enthalten weder Titelleiste, System-Button, Min-/Max-Button noch Windows-spezifische Menüs. Die Bedienung der Benutzeroberflächen erfolgt über Windows-Maus-Botschaften, die je nach installiertem Treiber von einer Maus, einem Trackball, einem Touchscreen oder weiteren Eingabeperipherien generiert werden. Alle Objekte sind ausreichend groß, um bei einer Bildschirmauflösung von 640×480 Pixeln (Windows-Minimum) und einem 15” Touchscreen eine fehlerfreie Bedienung zu gewährleisten.

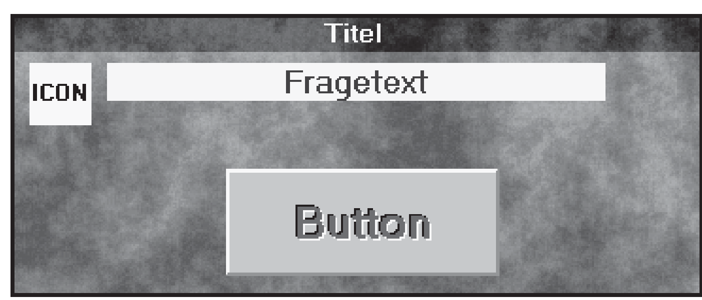
Die Benutzeroberflächen sind in drei Bereiche unterteilt:

1. Initialisierung (nur für den Anbieter)
2. Kiosk (für den Kunden)
3. Kommunikation

Der “Anbieter” entspricht dem Informationsgeber, der “Kunde” dem Informationsempfänger.

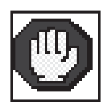
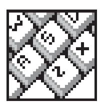
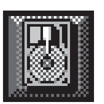
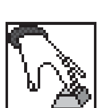
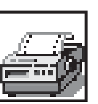
Beiden Oberflächen stehen zwei allgemein zugängliche Fenster zur Verfügung.
Das Rückfragefenster (Request) dient der Information und Programmsteuerung durch den Anwender. Es enthält mindestens einen Button, einen Titel und einen Informations- bzw. Fragetext. Angezeigte Buttons sind mit gleichen Abstand zentriert ausgerichtet.

System-Rückfragen



Elemente des Requestfensters

Jedem Request kann ein Icon je nach Sinn zugeordnet werden. Dazu stehen folgende zur Auswahl:

Achtung Eingabe Speichern Betätigen Drucken

Ein Request ist ebenfalls ein überlagerndes Fenster und legt andere KIOSK-Funktionen lahm.
Die Betätigung eines Buttons schließt den Request und aktiviert wieder das überlagerte Fenster. Die Nummer des betätigten Buttons wird über ein window-tag zurückgegeben.

Texteingaben

Für Texteingaben wird ein Fenster als Tastaturersatz geladen, welches die Eingabe von Ziffern, Versalien, den Umlauten Ä,Ö,Ü, Leerzeichen, Punkten und Bindestrichen gestattet. Jeweils das letzte Zeichen läßt sich löschen. Das Fenster wird über einen Schließbutton vom Bildschirm genommen und das Eingabeergebnis über ein window-tag zurückgegeben. Das Ergebnis der Eingabe ist in einer Ausgabezeile sichtbar. Die Position, an der das nächste eingegebene Zeichen ausgegeben wird, ist durch einen mittig stehenden Punkt gekennzeichnet.



Die Tastatur

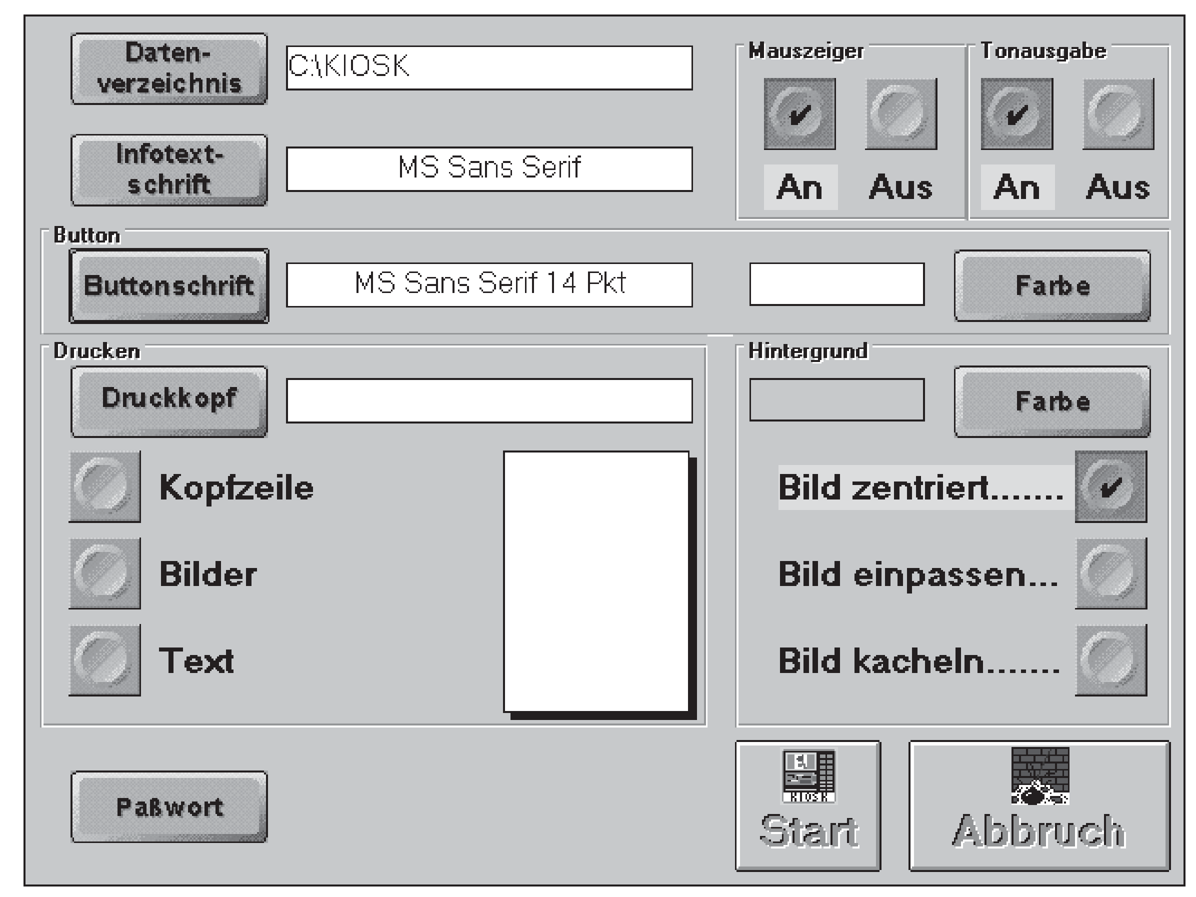
Die Anordnung der Buttons innerhalb des Fensters erfolgt nach der Norm der deutschen Schreibmaschinen-Tastatur. Das Fenster ist immer kleiner als 640×480 Pixel.

3.1.2.1 Initialisierung

Dem Anbieter steht ein Fenster zur Anpassung grundlegender Einstellungen zur Verfügung. Dieses ermöglicht über die Betätigung von entsprechend gekennzeichneten Buttons eine Initialisierung folgender Optionen:

- **Datenverzeichnis**
- **Infotextschrift**
- **Buttonschrift**
- **Buttonfarbe**
- *Hintergrundeinstellung:*
  - **Farbe**
  - **Bild zentriert**
  - **Bild einpassen**
  - **Bild gekachelt**
- **Druckkopf-Datei**
- *Druck-Einstellungen:*
  - **Kopfzeile**

Bilder
Texte
Mauszeigereinstellung:
An
Aus
Tonausgabeeinstellung:
An
Aus
Paßwort



Das Initialisierungsfenster mit seinen Standardeinstellungen

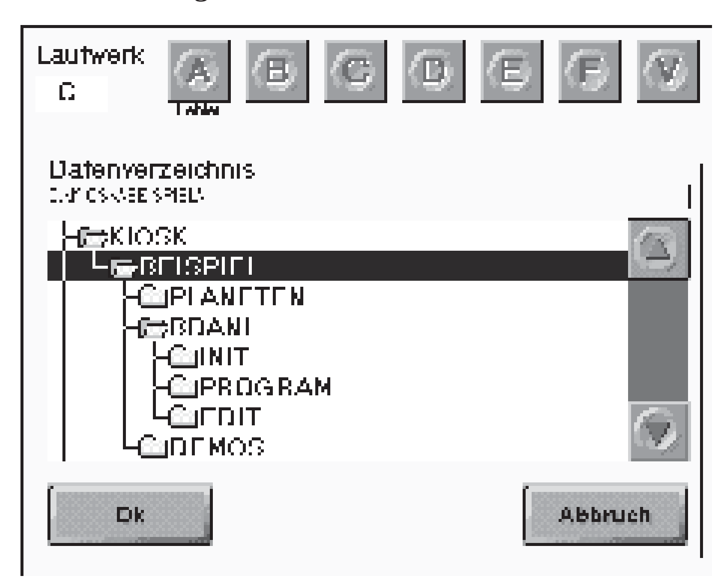
Allgemeines zu Dialogen

Zur Initialisierung stehen Dialogfenster zur Verfügung, die einen OK-Button zur Eingabebestätigung und einen Abbruch-Button zum Verwerfen der Eingaben enthalten. Eine Betätigung dieser Buttons schließt in jedem Fall den Dialog. Sind im Dialog Listen enthalten, so läßt sich ein Eintrag mittels Scrollbuttons auswählen. Bei dem Laden eines Dialogfensters, wird der Eintrag markiert, welcher der aktuelle Einstellung entspricht.

Datenverzeichnis wechseln

Für den Datenverzeichniswechsel erhält der Anbieter ein Dialogfenster, welches eines der installierten Laufwerke per Button zur Auswahl zuläßt, und ein Listen-

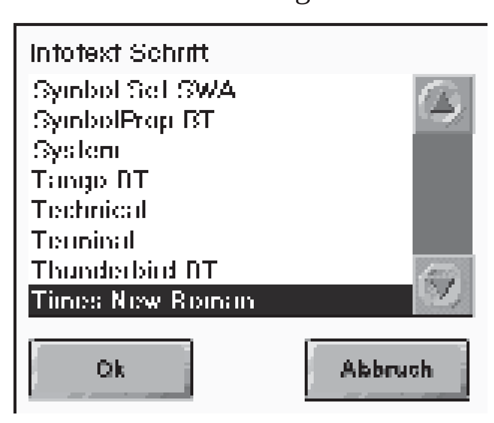
feld enthält, das den gesamten Verzeichnisbaum des gewählten Laufwerkes anzeigt. Ist ein Laufwerk nicht ansprechbar, wird der entsprechende Button mit “**Fehler”** gekennzeichnet und das nächste ansprechbare Laufwerk ausgewählt.



Dialog zum Datenverzeichniswechsel

Infotextschrift wechseln

Die im Textausgabebereich des Kiosks verwendete Schrift wird über den Dialog *Infotextschrift* angepaßt. Alle unter Windows verfügbaren Druckerschriften werden in einem Listenfeld dargestellt.

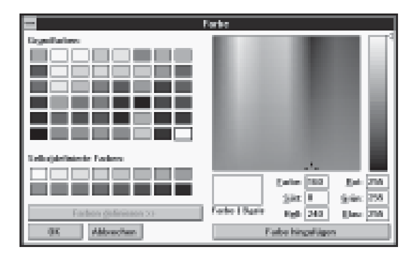


Dialog zum Infotextschriftwechsel

Buttonfarbe ändern

Die im Kiosk dargestellten Buttons erhalten eine individuelle Grundfarbe, die über den Windows-Standarddialog “*ChooseColor*” ausgewählt wird. Die Helligkeitsunterschiede für einen dreidimensionalen Effekt werden wie folgt berechnet:

Hell:	Rot, Grün und Blau-Anteil ×	133 %
Halbdunkel	Rot, Grün und Blau-Anteil ×	80 %
Dunkel:	Rot, Grün und Blau-Anteil ×	67 %

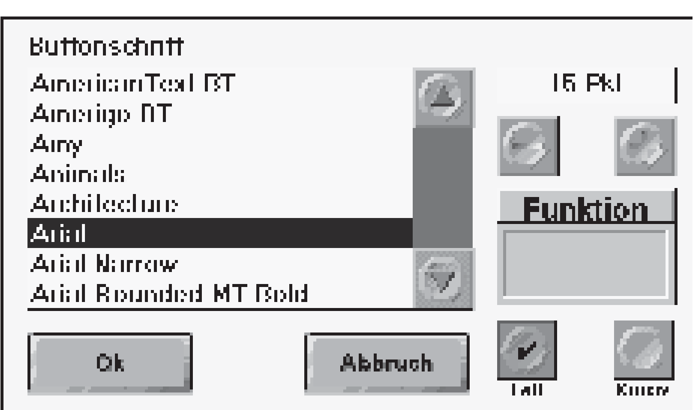


Dialog zur Farbauswahl

Mit “Hell” und “Halbdunkel” werden die Buttonränder schattiert. Die Bildausgabefläche wird mit “Dunkel” unterlegt.

Buttonschriftartwechsel

Die Schriftattribute der Buttons werden über den Dialog *Buttonschrift* eingestellt. Alle Windows-Bildschirmschriften werden in einem Listenfeld ausgegeben. Die Schriftgröße ist in 1-Punkt-Schritten über zwei Plus- und Minusbuttons änderbar. Die Schrift kann sowohl optional fett oder kursiv gewählt werden. Die aktuellen Einstellungen werden zur Kontrolle in einem im Dialog integrierten Musterbutton dargestellt.



Dialog zum Ändern der Buttonschriftattribute

Hintergrunddarstellung ändern



Die Auswahlbuttons des Hintergrundmodus

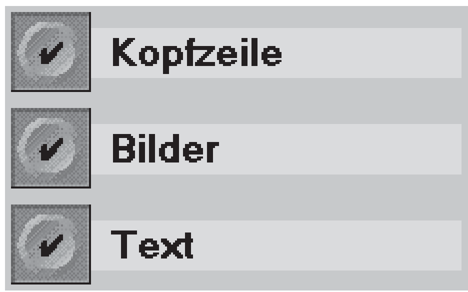
Bei dem Wechsel in ein neues Verzeichnis wird dieses auf das Vorhandensein einer BACKGR.BMP als gültige Windows-Bitmap-Datei geprüft. Ist keine vorhanden, werden alle darüber liegenden Verzeichnisse durchsucht. Ist ein Bild vorhanden, wird dieses in einem der drei Darstellungsmodi **zentriert**, **eingepaßt** oder **gekachelt** in den Farben der Systempalette (ggf. gerastert) dargestellt. Es kann und muß immer nur ein Modus aktiv sein. Der ausgewählte Modus wird über einen Haken im Button dargestellt. Ist kein Bild vor-

handen, oder der Hintergrund nicht vollständig vom Bild bedeckt, wird dieser in der unter Hintergrund definierten Farbe gefüllt. Die Farbauswahl erfolgt ebenfalls über den Windows-Standarddialog "*ChooseColor*".

Druckoptionen

Alle im Kiosk dargestellten Grafiken und Texte können ggf. auf Wunsch des Kunden ausgedruckt werden. Welche Dateien druckbar sein sollen, wird über entsprechende Buttons gewählt. Jeder Button ist in der Auswahl unabhängig vom Zustand der anderen. In einem Beispielbild wird der aktuelle Zustand dargestellt.

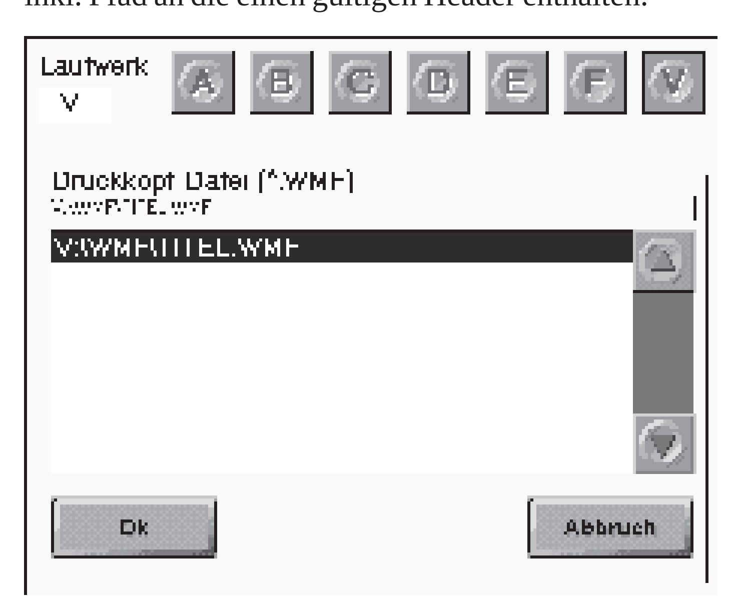




Auswahlmöglichkeit der Druckoptionen

Bei jeder Betätigung eines Buttons wechselt dieser seinen Zustand. In der Kopfzeile eines Blattes kann ein anbieterspezifisches Windows-Metafile ausgedruckt werden. Diese Datei wird über den Dialog *Druckkopf-Datei* ausgewählt. Der Dialog läßt eine Auswahl eines der installierten Laufwerke per Button zu. Ist ein Laufwerk nicht ansprechbar, wird der entsprechende Button mit "**Fehler**" gekennzeichnet und das nächste ansprechbare Laufwerk ausgewählt. Ein Listenfeld zeigt alle auf dem gewählten Laufwerk befindlichen WMF-Dateien inkl. Pfad an die einen gültigen Header enthalten.

Wechseln der Druckkopfdatei



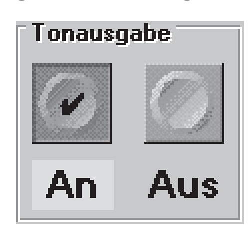
Dialog zum Wählen des Druckkopfes

Einstellung des Mauszeigers



Da das Kiosk-System für einen Betrieb mit einem Touchscreen vorgesehen ist, kann der Mauszeiger über entsprechende Buttons ausgeschaltet werden. Das Abschalten des Cursors erfolgt mit dem Start des Kiosks. Mit dem Beenden des Programmes wird dieser wieder sichtbar. Während der Initialisierung wird der Mauszeiger durch den Windows-Sperrzeiger dargestellt ⊘ .

Einstellung der Tonausgabe



Die Tonausgabe ist ebenfalls über entsprechende Buttons an- bzw. abschaltbar.

Paßwortvorgabe



Eingebe eines Paßwortes

Für die Editierung eines Paßwortes wird das Tastaturfenster geladen. Der eingegebene Text wird symbolisch durch einen Kreis pro Zeichen dargestellt. Nach der Bestätigung des Paßwortes muß eine zweite Eingabe erfolgen, die der ersten übereinstimmt. Ist dies nicht der Fall, wird das Paßwort gelöscht und der Anbieter zur Neueingabe aufgefordert. Das Paßwort wird verschlüsselt in der nachfolgenden Codierung gespeichert.
Die Codierung erfolgt in Abhängigkeit der Position und des ASCII-Codes eines Zeichens. Jedes Zeichen wird durch ein neues ersetzt. Der ASCII-Code des neuen Zeichens errechnet sich aus:

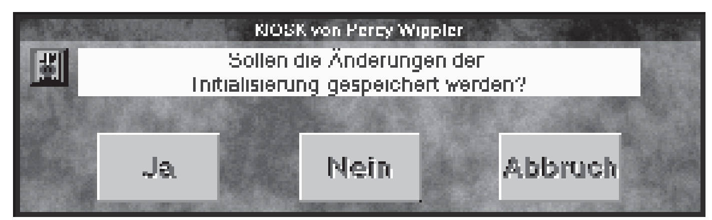
$$\mathbf{(ASCII_{\mathit{Original}} - 32) \times (4 - Pos._{\mathit{Zeichen}}\ MOD\ 3)}$$

Die Rückrechnung eines codierten Zeichens erfolgt über:

$$\mathbf{ASCII_{\mathit{VERSCHLÜSSELUNG}}\ DIV\ (4 - Pos._{\mathit{Zeichen}}\ MOD\ 3) + 32}$$

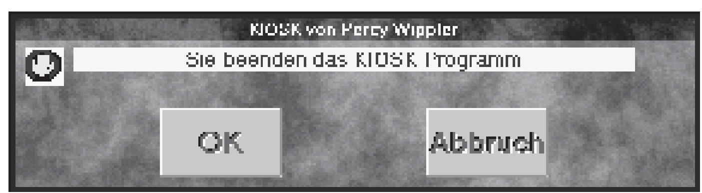
Ist ein Paßwort vorgegeben, muß dieses zum Programmstart eingegeben werden. Bei einer Fehleingabe erfolgt ein Request, der die Auswahl zum Programmabbruch oder zu einem Neuversuch vorgibt.

Beim Beenden der Initialisierung können Änderungen wahlweise gespeichert oder ignoriert werden. Dies geschieht über einen entsprechenden Request.



Request bei vorgenommen Initialisierungsänderungen

Bei einer Betätigung von *Abbruch* wird das Initialisierungsfenster nicht geschlossen. Geänderte Initialisierungen sind auch ohne Speichern für einen folgenden Start des Kiosks gültig.
Der Kiosk wird direkt aus der Initialisierung gestartet, wenn der Button *Start* (des Initialisierungsfensters) betätigt wird. Wird *Abbruch* betätigt, wird dies über einen Request abgefangen.



Request bei Programmende

Nach dem Laden der Software und einer ggf. korrekten Paßworteingabe läßt ein Request die Initialisierung, den direkten Kiosk-Start oder einen Programmabbruch zu.



Request bei Programmstart

3.1.2.2 Kiosk

Nach dem Start des Kiosks erhält der Kunde ein auf Vollbild vergrößertes Fenster. Dieses ist mit Objekten versehen, die wie folgt angeordnet sind:

Bildfeld:
max. Breite × *Höhe* = 5/8 × 1/2 der Fenstergröße
Ausrichtung: oben
Animationsfeld:
max. Breite × *Höhe* = 5/8 × 1/2 der Fenstergröße
Ausrichtung: oben
Rahmen:
umrahmt Bild- und Animationsfeld
Infozeile:
Breite = 5/8 der Fensterbreite
Höhe = Buttonschrifthöhe
Ausrichtung: unten
Textfeld:
Breite = 5/8 der Fensterbreite
Höhe = Fensterhöhe - Rahmenhöhe - Infoz.höhe
Ausrichtung: zwischen Rahmen und Infozeile
Scroll-Leiste:
Breite = 40 Pixel
Höhe = Textfeldhöhe
Ausrichtung: links bzw. rechts des Textfeldes
Themenbuttons:
Breite × *Höhe* = 3/20 × 2/15 der Fenstergröße
Ausrichtung: in jeweils einer in zwei Spalten geteilten Dekade am oberen Fensterrand
Steuerbuttons:
Breite × *Höhe* = 3/20 × 1/15 der Fenstergröße
Ausrichtung: sechs in zwei Spalten am unteren Fensterrand mit den Bezeichnungen:

Zurück	**Mehr**
Übersicht	**Suchen**
Drucken	**Kommentar**

Händerbuttons:
Breite × *Höhe* = 140 × 140 Pixel
Ausrichtung: Zwei in der Mitte des Textfeldes
Diese Buttons sind ausschließlich in der Übersicht (Datenverzeichnis) sichtbar.





Die Händerbuttons für eine Anordnung gemäß : Links- / Rechtshänder

Die Ausgabeobjekte befinden sich auf der einen Bildschirmseite, die Buttons auf der anderen. Über die

Händerbuttons kann der Kunde die Seiten dieser Bereiche vertauschen.
Alle Objekte passen sich der Bildschirmauflösung an. Sie haben zueinander einen Abstand von 1,1% der Fensterbreite.



Objekte des Kiosks

Die Schriftart der Informationszeile und Kontrollbuttons ist mit der Schrift der Themenbuttons identisch.
Objekte sind nur dann sichtbar, wenn Ihnen eine Funktion zugeordnet ist.

Die Funktionen der Buttons

Die Betätigung eines Themenbuttons hat die Ausführung der in der INFO.INI zugewiesenen Befehle zur Folge.

<Zurück>

1. Fall: Erste Themenbuttondekade sichtbar:
Zurück geht den gesamten Weg des Kunden über die von ihm gewählten Themen mit einem Schritt pro Betätigung zurück. Es werden maximal 30 Positionen gespeichert.
2. Fall: Nicht erste Themenbuttondekade sichtbar:
Zurück zeigt die vorherige Dekade an.

<Mehr>

Sind mehr als zehn Themenbuttons vorhanden, zeigt *Mehr* die jeweils folgende Dekade an.

<Übersicht>

Übersicht wechselt in das Ausgangs- (Daten-) Verzeichnis. Der Button ist nur dann sichtbar, wenn eine anderes als das Datenverzeichnis aktuell ist.

<Suchen>

Suchen öffnet einen Dialog, der ein Thema aus einer Liste auswählen läßt und diesen direkt angehen kann. (Siehe nächste Seite)

<Drucken>

1. Fall: Drucker-Spooler leer:

- Akt. Grafik und Text werden gespoolt.
- Ein Request fragt, ob jetzt oder später gedruckt werden soll.

2. Fall: Drucker-Spooler nicht leer:

- Ein Request fragt ob gedruckt oder neue Daten gespoolt werden sollen. Zum Spoolen wird der 1. Fall ausgeführt

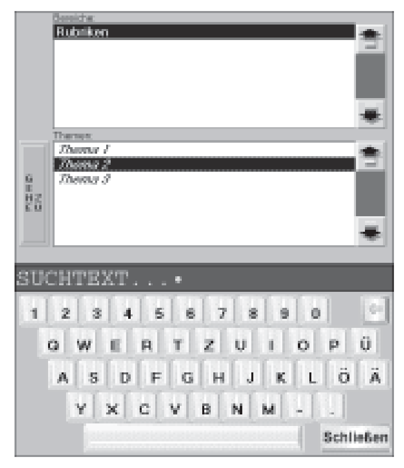
<Kommentar>

Kommentar gestattet dem Kunden über das Tastaturfenster einen beliebigen Kommentar zu geben, der in einer Textdatei **VORSCHLG.TXT** im ANSI-Format gespeichert wird.

Je nach Arbeitsvorgang oder Zustand des Kiosks werden in der Infozeile folgende Texte ausgegeben:

Warten Sie bitte einen Moment ...
<Mehr> zeigt Ihnen weitere Themen
<Zurück> zeigt Ihnen vorherige Themen
<Zurück> zeigt Ihnen Ihr letztes Thema
Wählen Sie bitte ein Thema
Verschieben Sie den Text mit den Pfeiltasten
Lade Daten zum Thema: *Themenbuttontitel*
Derzeitige Auswahl: *Themenbuttontitel*
Ordnen Sie die Schaltflächen gemäß Ihrer Gewöhnung an
Bitte machen Sie Ihre Eingabe, und beenden Sie mit <Schließen>
Rückfrage! Wählen Sie bitte eine der Antworten
Die Daten werden gedruckt ...
Die Druckdaten werden erstellt ...

Nach fünf vergangenen Minuten ohne Betätigung eines Buttons schaltet der Kiosk automatisch in die Übersicht zurück.



Der Dialog zur Index-Suche

Für die Suche eines bestimmten Themas, besteht die Möglichkeit über die Tastatur einen Bereich einzugeben oder einen Bereich aus einer Liste zu wählen. Zu jedem gewählten Bereich werden in einer zweiten Liste die entsprechenden Themen dargestellt. Mit dem Button **Gehezu** wird das gewählte Thema im Kiosk präsentiert. **Schließen** bricht die Suche ohne Änderungen der aktuellen Information ab. Eine Texteingabe wählt bei jedem eingegebenen Zeichen in der Bereichliste den wahrscheinlichsten Eintrag aus.

3.1.2.3 Kommunikation

Im Kiosk-System besteht die Möglichkeit zur Kommunikation mit anderen Systemen, z.B. einem Kassenprogramm, über die Schnittstelle einer *Dynamic Link Library* (DLL). Diese kann vom Anbieter selbst erstellt werden und muß folgenden Konventionen entsprechen:

1. Sie muß den Namen **KIOSKDRV.DLL** erhalten
2. Es müssen mindestens zwei Funktionen oder Prozeduren namens

GetItemPrice und **GetItemInvent**

als exportierbar deklariert sein. Beiden Routinen werden zwei Pointer auf Null-terminierte Strings übergeben. Der erste zeigt auf einen String, der die Artikelkennung enthält, der zweite zeigt auf einen 255 Zeichen großen Puffer, der das Ergebnis der Kommunikation aufnimmt.
Die Kommunikationsbefehle sind als Sonderkommandos in MS-Rich-Text-Dateien zu integrieren. Im Fall eines Kommunikationsfehlers soll “*Fehler*” im Text angezeigt werden. Als Standard liegt eine KIOSKDRV.DLL dem Kiosk-System bei, die den Text “*Bitte erfragen*” ausgibt.

3.2 Datenformate

Das Kiosk-System kann folgende Dateiformate erkennen und umsetzen:

3.2.1 ANSI-Format

Entspricht dem genutzten Zeichensatz von Windows. Keine weiteren Deklarationen enthalten.

3.2.2 MS-Rich-Text-Format

Eine RTF-Textdatei Liegt im ANSI-Format vor. Die Kennung erfolgt über die ersten sechs Byte, die identisch mit '**{\rtf1**' sein müssen. Jeder Formatbefehl wird immer durch einen Backslash "**"** eingeleitet. Alle Formatierungsabschnitte eines Textes sind durch geschweifte Klammern **"{}"** begrenzt. Eine RTF-Datei beginnt immer mit **{** und endet immer mit **}**. Folgende Formatierungsbefehle kommen für die Textausgabe zur Wirkung:

\rtf*n* .... *identifiziert die Datei als RTF-Datei und gibt die Versionsnummer n des benutzten RTF-Standards an.*

Farbattribute

{**\colorttbl;**
\red*r***\green***g***\blue***b***;**
⋮
} .... *erzeugt eine Farbtabelle für die Textausgabe. Sie besteht aus mindestens einer Farbdefinition. Jede Farbdefinition besteht aus je einer* **\red***-,* **\green-** *und* **\blue-** *Anweisung mit abschließendem Semikolon "***;***". Für r, g und b sind Intensitätswerte im Bereich 0 bis 255 zulässig.*

\cf*n* .... *setzt die Textausgabefarbe auf den Tabellenwert n. Der erste Eintrag einer definierten Farbtabelle hat den Index Eins. Ist für n Null angegeben, wird die Untergrundfarbe gewählt.*

Absatzattribute	**\li***n*	*setzt den linken Einzug für alle Zeilen auf n Twips (= n/1440 Zoll).*
	\fi*n*	*setzt den linken Einzug für die erste Zeile eines Absatzes auf n Twips.*
	\ri*n*	*setzt den rechten Einzug für alle Zeilen auf n Twips.*
	\ql	*richtet Text linksbündig aus.*
	\qr	*richtet Text rechtsbündig aus.*
	\qc	*zentriert Text (mittig).*
	\pard	*setzt alle Absatzattribute auf Standardwerte:* *- Ausrichtung .... linksbündig.* *- Erstzeileneinzug. Null.* *- Linker Einzug .... Null.* *- Rechter Einzug .. Null.*
Zeichenattribute	**\b**	*schaltet Fettdruck ein.*
	\b0	*schaltet Fettdruck aus.*
	\i	*schaltet Kursivdruck ein.*
	\i0	*schaltet Kursivdruck aus.*
	\ul	*schaltet Unterstreichen ein.*
	\ul0	*schaltet Unterstreichen aus.*
	\strike	*schaltet Durchstreichen ein.*
	\strike0	*schaltet Durchstreichen aus.*
	\fs*n*	*setzt die Größe einer Schrift in n halben Punkten.*
	\plain	*setzt alle Zeichenattribute auf Standardwerte:* *- Fettdruck .... aus.* *- Kursivdruck .... aus.* *- Unterstreichen ... aus.* *- Durchstreichen .. aus.* *- Schriftgröße .... 10 Pkt.*
Ausgabeattribute	**\par**	*markiert das Ende eines Absatzes und führt einen Zeilenumbruch aus.*
	\line	*bricht die aktuelle Zeile um, ohne den Absatz zu beenden.*
	\'*hh*	*wandelt die gegebene Hexadezimalzahl hh in ein Zeichen und fügt es in den Text ein. Das Aussehen des Zeichens hängt vom gesetzten Zeichensatz ab.*
	\tab	*fügt einen Tabulator ein.*



Alle weiteren Steuerbefehle werden vom Programm ignoriert. Spalten, Tabellen und Grafiken sind nicht darstellbar. Soll im Text ein Backslash oder eine geschweifte Klammer ausgegeben werden, so muß diesem Zeichen ein weiterer Backslash vorangehen: “****” oder “**\{**“.

Nicht im RTF enthaltene Steuerbefehle

KIOSK-spezifische Steuerbefehle, werden durch ein “@” eingeleitet. Dazu gehören:

Farbattribute

@**cp***n* .... *setzt die Untergrundfarbe auf den Tabellenwert n. Der erste Eintrag einer definierten Farbtabelle hat den Index Eins. Ist für n Null angegeben, wird die Untergrundfarbe auf den Standardwert gesetzt.*

@**cb***n* .... *setzt für die Textausgabe die Hintergrundfarbe auf den Tabellenwert n. Der erste Eintrag einer definierten Farbtabelle hat den Index Eins. Ist für n Null angegeben, wird die Untergrundfarbe gewählt.*

Artikeldaten

@**Preis***artnr* .... *ermittelt über die Link-Library KIOSKDRV.DLL den aktuellen Preis eines Artikels. Dieser wird über seine Kennung artnr identifiziert.*

@**Bestand***artnr* ...... *ermittelt über die Link-Library KIOSKDRV.DLL den aktuellen Bestand eines Artikels, der über artnr identifiziert wird.*

3.2.3 Windows-Bitmap

Eine BMP-Datei enthält einen File-Header (14 Byte), einen Bitmap-Info-Header (40 Byte), eine Farbtabelle (beliebiges Vielfaches von 4 Byte) und einen Datenbereich.

Offset	Bytes	Name	Bedeutung
00H	2	bfType	File ID ('BM')
02H	4	bfSize	Filelänge in Byte
06H	4	-----	reserviert (muß 0 sein)
08H	2	-----	reserviert (muß 0 sein)
0AH	4	bfOffs	Offset Datenbereich

Aufbau des Windows-BMP-Headers

Offset	Bytes	Name	Bedeutung
0EH	4	biSize	Länge des info-Headers in Byte
12H	4	biWidth	breite der Bitmap in Pixeln
16H	4	biHeight	Höhe der Bitmap in Pixeln
1AH	2	biPlanes	Zahl der Farbebenen (muß 1 sein)
1CH	2	biBitCount	Zahl der Bits pro Pixel (Farbtiefe)
1EH	4	biCompr	Typ der Komprimierung
22H	4	BISizeIm	Bildgröße in Byte
26H	4	biXPixel/Inch	horizontale Auflösung
2AH	4	biYPixel/Inch	vertikale Auflösung
2E	4	biColorUsed	Zahl der benutzten Farben
32H	4	biColorImp.	Zahl der wichtigen Farben
			RGB_QUAD
26H	*n×4*	rgbBlue rgbGreen rgbRed rgbRes	Ffarbdefinition für n Farben mit: 1 Byte Intensität Blau 1 Byte Intensität Grün 1 Byte Intensität Rot 1 Byte reserviert ...

Aufbau des Windows-BMP-Info-Blocks

3.2.4 Windows-Metafile

Windows-Metafiles können mit und ohne Header versehen sein. WMF-Dateien ohne Header werden nicht umgesetzt, da keine Informationen über die Proportionen der Grafik bezüglich Breite und Höhe bestehen.
Der Header besteht aus 22 Byte und enthält folgende Informationen:

Offset	Bytes	Name	Bedeutung
00H	4	key	Kennung (muß 9AC6CDD7h sein)
04H	2	hmf	reserviert (muß 0 sein)
06H	8	bbox	kleinstes umschließendes Rechteck (x1,y1,x2,y2) in 4 Word
14H	2	units/inch	Einheiten für bbox
16H	4	res	reserviert (muß 0 sein)
20H	2	Checksum	XOR-Ergebnis der ersten 10 Words

Aufbau des Windows-Metafile-Headers

3.2.5 Video-For-Windows

Die Ausmaße und die Identifikation einer Animation können aus den ersten 24 Byte gelesen werden.

Offset	Bytes	Name	Bedeutung
00H	4	riff	Formatkennung (muß 'RIFF' sein)
04H	4	size	Dateigröße in Bytes
08H	8	type	Dateityp (muß 'AVI LIST' sein)
10H	4	width	Breite der Animation in Pixeln
14H	4	height	Höhe der Animation in Pixeln

Aufbau des Video-For-Windows-Headers

3.2.6 Wave-Dateien

Die Identifikation einer Klangdatei kann aus den ersten 16 Byte gelesen werden.

Offset	Bytes	Name	Bedeutung
00H	4	riff	Formatkennung (muß 'RIFF' sein)
04H	4	size	Dateigröße in Bytes
08H	7	type	Dateityp (muß 'WAVEfmt' sein)

Aufbau des Wave-Headers

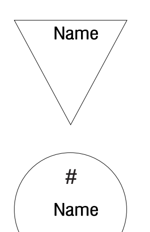
3.2.7 INI-Dateien

Initialisierungsdateien sind im ANSI-Format gespeichert. Sie werden für die Makro- und Indexerstellung verwendet. Eine Initialisierungsdatei kann mit der Windows-Funktion **GetPrivateProfileString** gelesen werden. Hierbei muß die INI-Datei folgenden Konventionen entsprechen.
Jede Bereichsbezeichnung ist von eckige Klammern eingeschlossen. Nachfolge Zuweisungen bestehen aus: Schlüsselwort, Gleichheitszeichen und Zuweisung.
In einer INI-Datei kann eine Bereichsbezeichnung wiederholt vorkommen, es ist aber immer nur die erste Zuweisung gültig.

3.3 Objekte und Bezeichner des Zustands-Aktions-Modells

ZA-Elemente

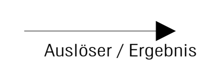
Elemente der methodischen Darstellung nach **ZA** sind:



Einstieg (Modulname)



Zustand



Übergang

Name

Aktion

Zustände und Aktionen werden mit Eins beginnend durchnummeriert und entsprechen den im Formular angegebenen Ebenennummerierungen. Gleiche Module in einem Diagramm enthalten identische Bezeichner. **Zustände**, sowie **Aktionen** können verfeinert werden. Verfeinerte Module werden schattiert dargestellt.
Pro Diagramm existiert genau ein **Einstieg**. Der Einstiegsname einer unteren Ebene entspricht dem verfeinerten Modulnamen aus der höheren Ebene. Module sind immer durch einen **Übergang** verbunden.

Das Formular

Das Formular enthält einen Rahmen und eine Fußzeile, in der der Name (Kontext) des Moduls, die Ebenentiefe, der Projektname, sowie das Datum der letzten Bearbeitung und der Bearbeiter einzufügen sind.

Kontext

Ebene:	*#.#.#...*	**Stand vom:**	*tt.mmm.jjjj*
Projekt:	*Projektname*	**Bearbeiter:**	

Die Ebenennummerierung ergibt sich aus der Nummerierung der nächst höheren Ebene, gefolgt von einem Punkt und der Nummer des verfeinerten Moduls. Globale Module, die von allen Programmodulen verwendet werden können, befinden sich auf der Ebene 0. Die Ebene des Systemmodells ist mit 1 gekennzeichnet. Von hier gehen weitere Funktionsmodelle aus.

Namensgebung

Ereignis-, Zustands- und Aktionsnamen sind zur Identifizierung mit bestimmten Kennungen versehen. Diese gehen dem Objektnamen voran. Je nach Objekt sind folgende Kennungen vorgesehen:

Objektidentifizierung

Objekt	**Kennung**
Fenster (Form)	**frm***Name*
Schaltfläche (Button)	**but***Name*
Textbereich	**txt***Name*
Grafikbereich	**pic***Name*
Animationsbereich	**anim***Name*
Rahmen	**frame***Name*
Listen	**lst***Name*
Dateien	**dat***Name*
Modi	**mod***Name*
Multimedia-Steuerung	**mci***Name*

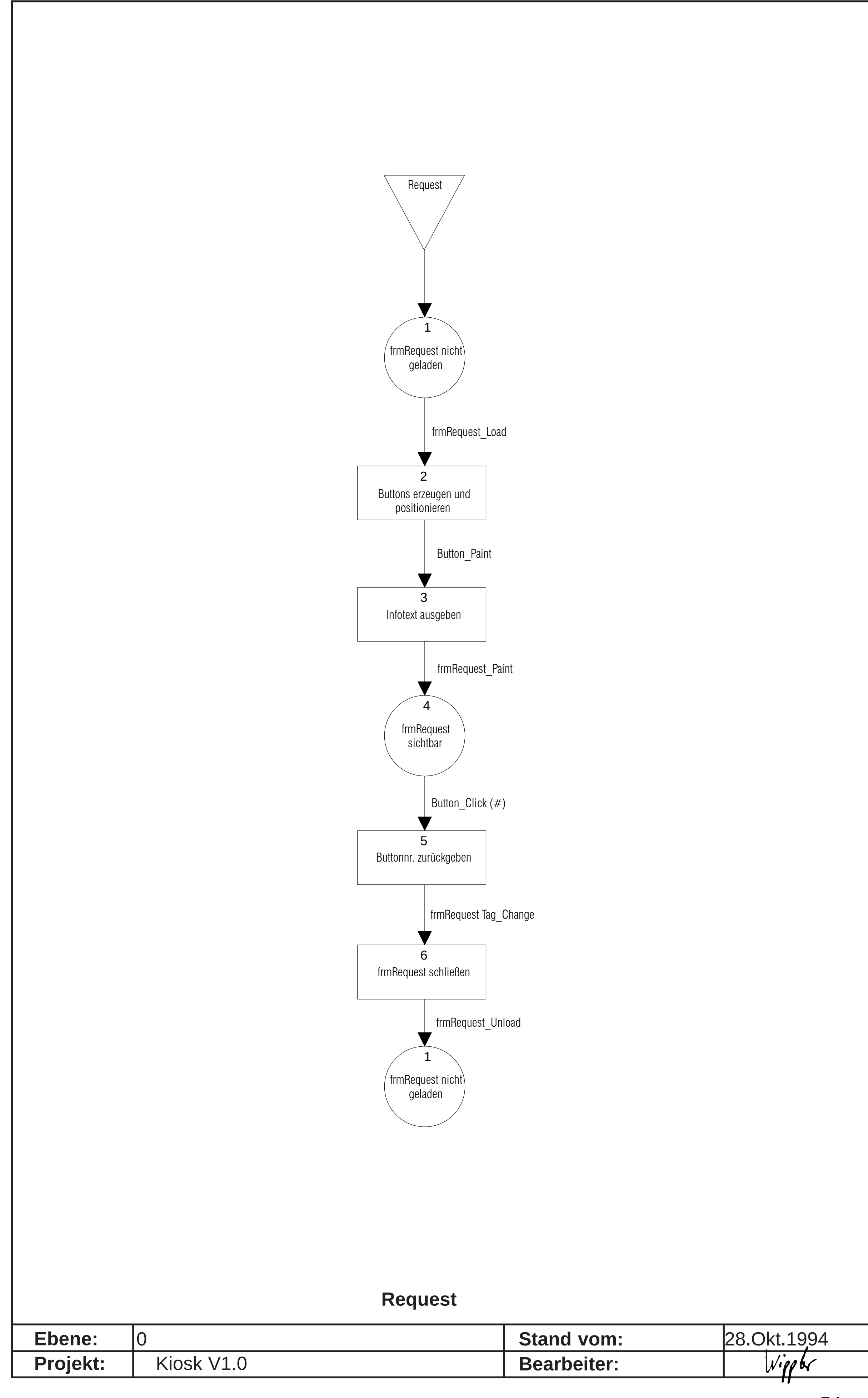
Ereignisse werden je nach Typ mit einem Unterstrich und folgenden Kennungen erweitert:

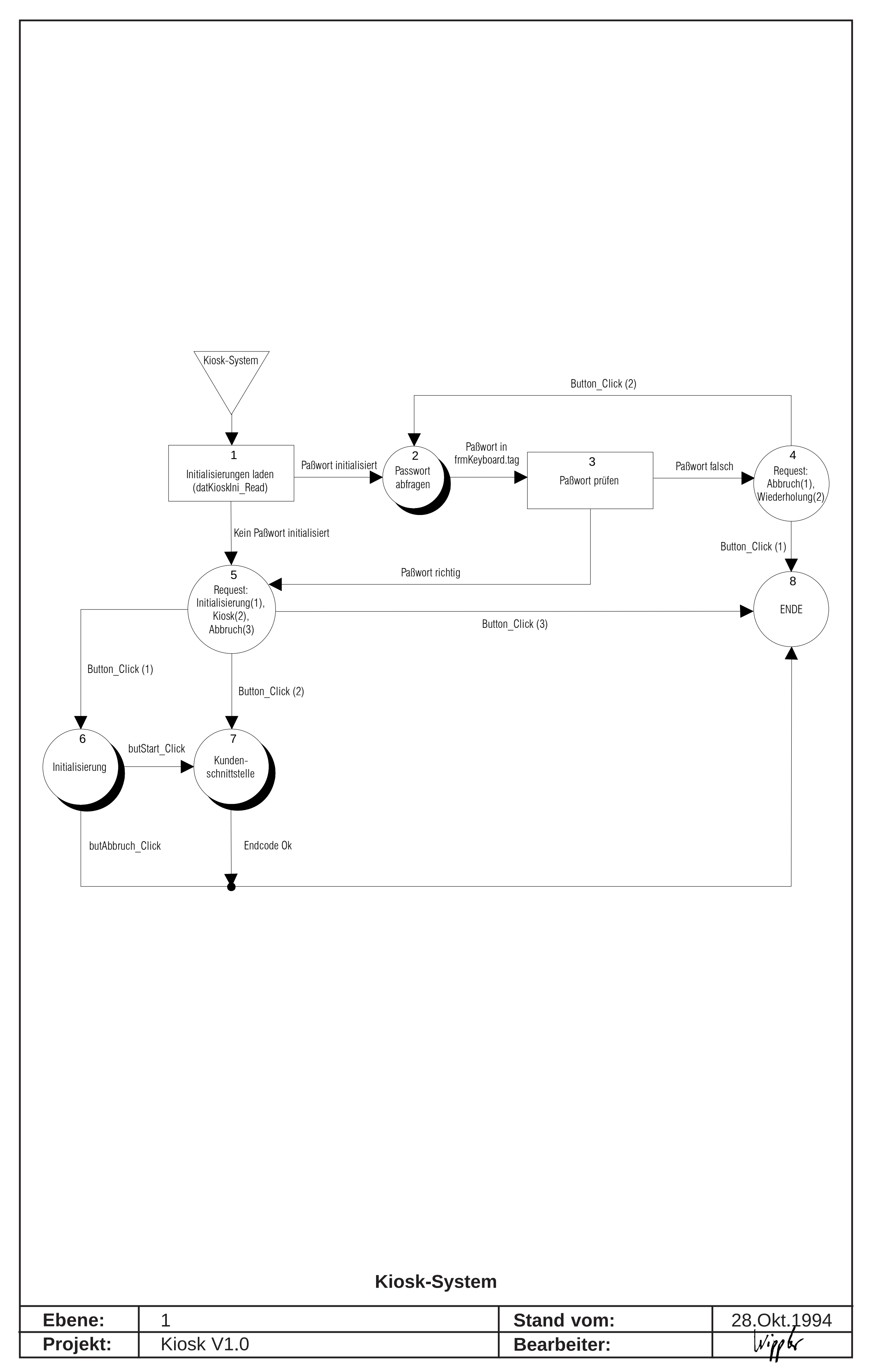
Ereignisidentifizierung

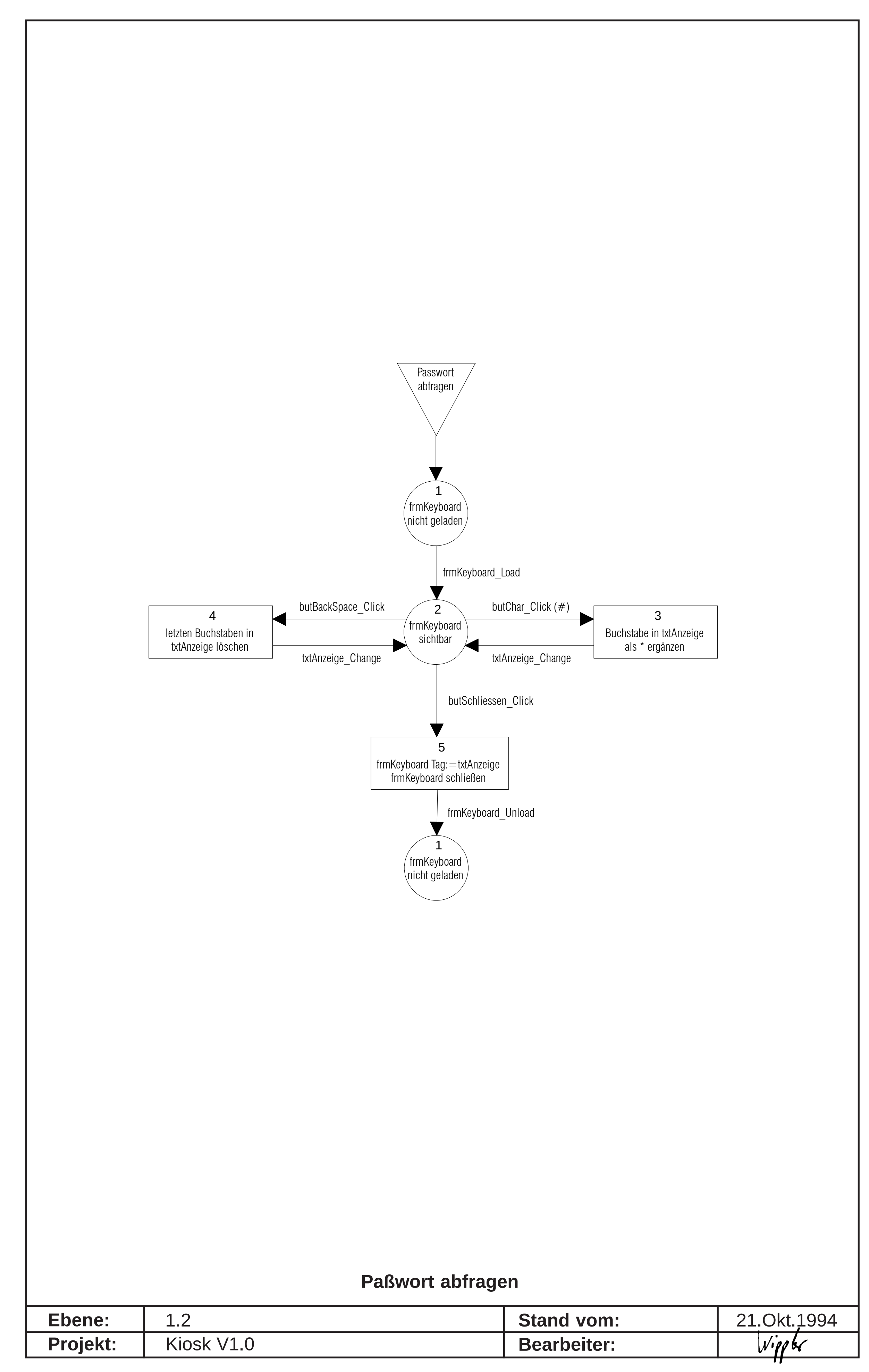
Ereignis	**Kennung**
Objekt laden	*Name***_Load**
Objekt entfernen	*Name***_Unload**
Objektdarstellung ändern	*Name***_Paint**
Mausklick (Touch)	*Name***_Click**
Scrollen	*Name***_Scroll**
Daten speichern	*Name***_Write**
Daten lesen	*Name***_Read**
Abspielen von Klängen und Animationen	*Name***_Play**
Abspielen beendet	*Name***_Done**
Inhalte ändern	*Name***_Change**

Objektfelder (mehrere Objekte eines Typs) werden durch nachfolgende Klammern gekennzeichnet “**()**”. Einem unbestimmten Feldelement wird ein Doppelkreuz hinzugefügt “**(#)**”. Ein bestimmtes erhält eine eindeutige Identifizierung z.B. eine Ziffer “**(1)”**.

3.4 System- und Funktionsmodelle

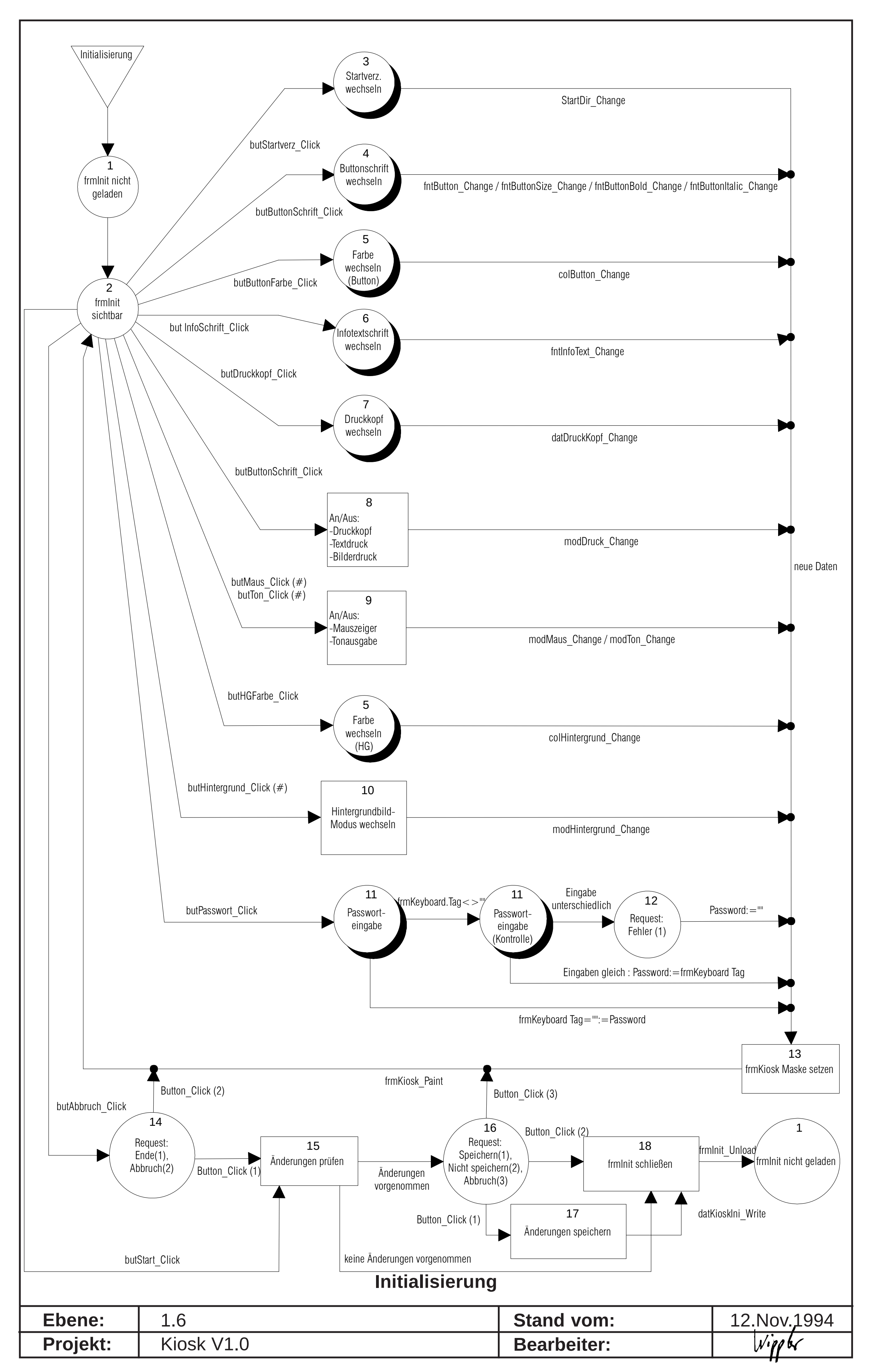

Request
1
frmRequest nicht geladen
frmRequest_Load
2
Buttons erzeugen und positionieren
Button_Paint
3
Infotext ausgeben
frmRequest_Paint
4
frmRequest sichtbar
Button_Click (#)
5
Buttonnr. zurückgeben
frmRequest Tag_Change
6
frmRequest schließen
frmRequest_Unload
1
frmRequest nicht geladen
Request
Ebene:
0
Stand vom:
28.Okt.1994
Projekt:
Kiosk V1.0
Bearbeiter:

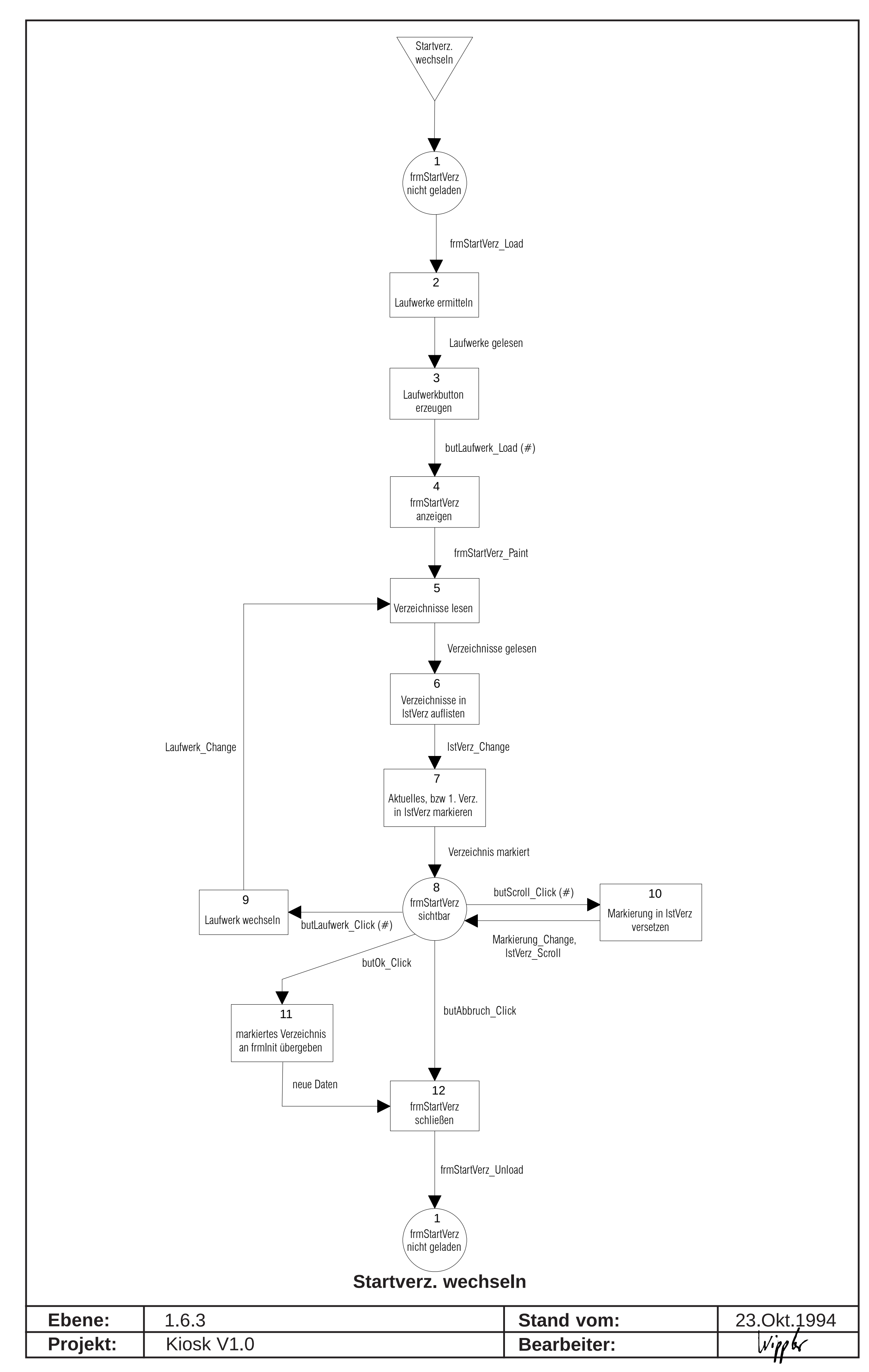

Kiosk-System
1
Initialisierungen laden
(datKioskIni_Read)
Paßwort initialisiert
2
Passwort
abfragen
Paßwort in
frmKeyboard.tag
3
Paßwort prüfen
Paßwort falsch
4
Request:
Abbruch(1),
Wiederholung(2)
Button_Click (2)
Button_Click (1)
Kein Paßwort initialisiert
Paßwort richtig
5
Request:
Initialisierung(1),
Kiosk(2),
Abbruch(3)
8
ENDE
Button_Click (3)
Button_Click (1)
Button_Click (2)
6
Initialisierung
butStart_Click
7
Kunden-
schnittstelle
butAbbruch_Click
Endcode Ok
Kiosk-System
Ebene:
1
Stand vom:
28.Okt.1994
Projekt:
Kiosk V1.0
Bearbeiter:



Paßwort abfragen

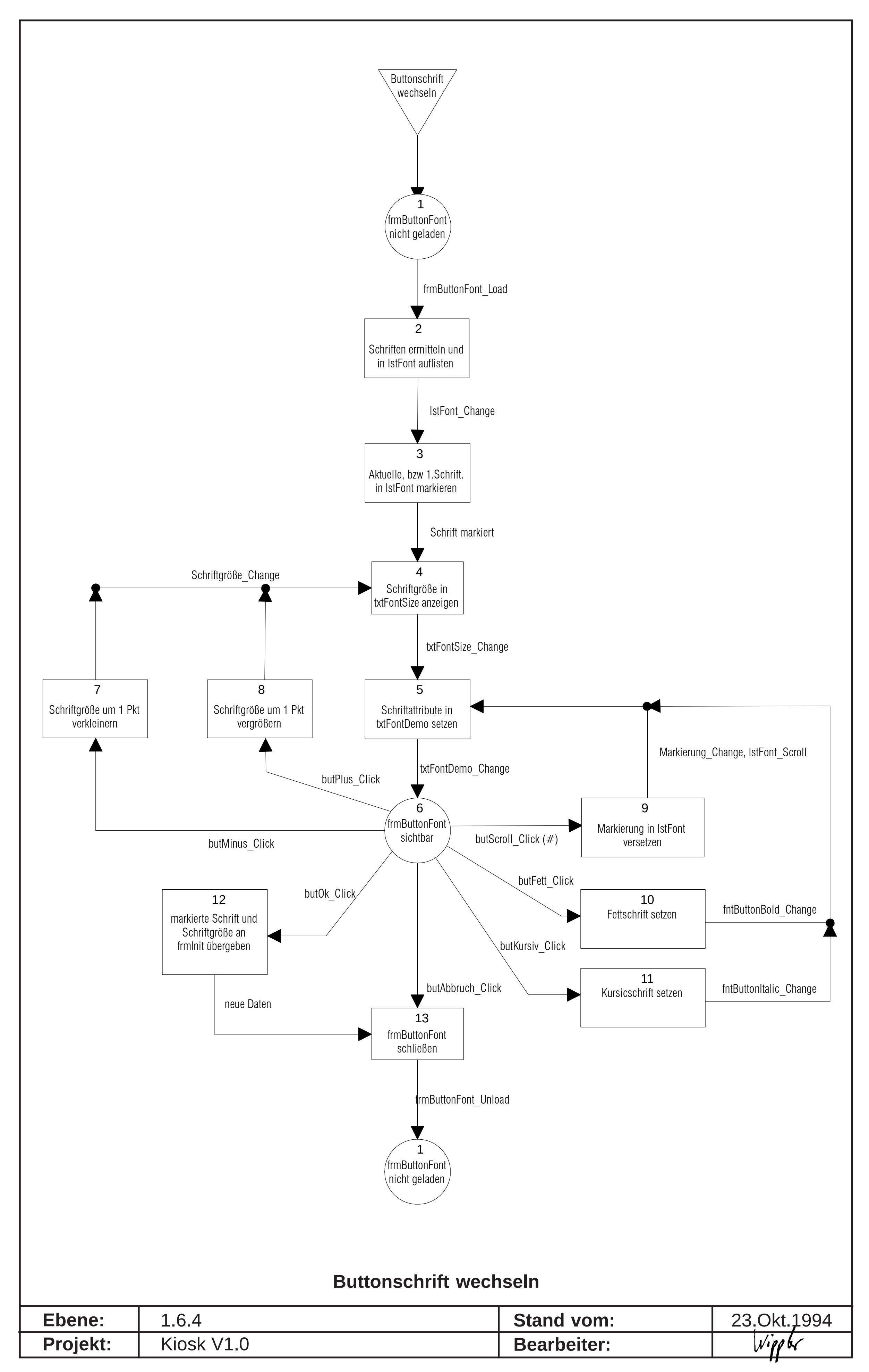
Ebene:	1.2	**Stand vom:**	21.Okt.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	Wippler

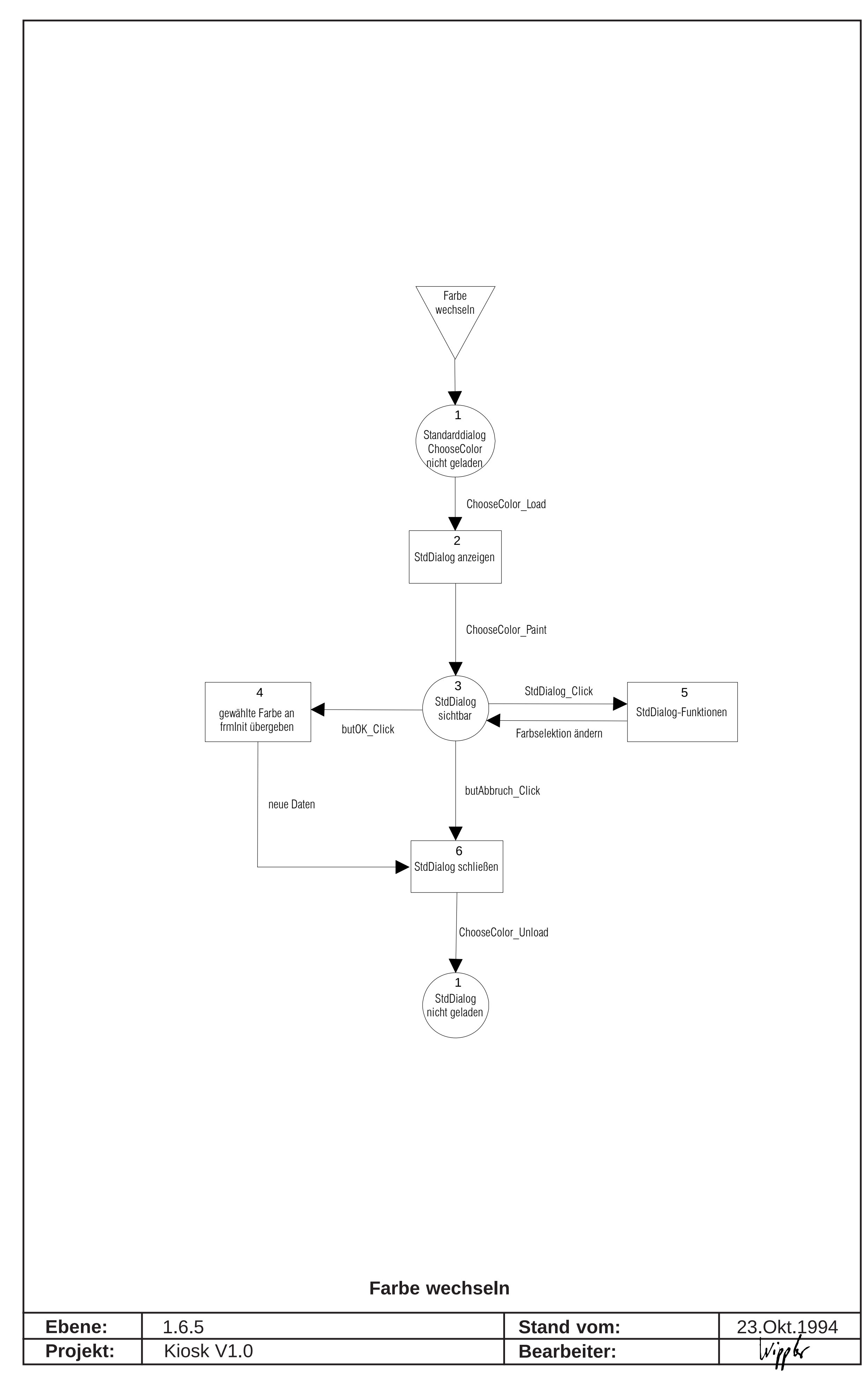

Initialisierung
1
frmInit nicht geladen
2
frmInit sichtbar
3
Startverz. wechseln
StartDir_Change
butStartverz_Click
4
Buttonschrift wechseln
fntButton_Change / fntButtonSize_Change / fntButtonBold_Change / fntButtonItalic_Change
butButtonSchrift_Click
5
Farbe wechseln (Button)
butButtonFarbe_Click
colButton_Change
6
Infotextschrift wechseln
but InfoSchrift_Click
fntInfoText_Change
butDruckkopf_Click
7
Druckkopf wechseln
datDruckKopf_Change
butButtonSchrift_Click
8
An/Aus:
-Druckkopf
-Textdruck
-Bilderdruck
modDruck_Change
neue Daten
butMaus_Click (#)
butTon_Click (#)
9
An/Aus:
-Mauszeiger
-Tonausgabe
modMaus_Change / modTon_Change
butHGFarbe_Click
5
Farbe wechseln (HG)
colHintergrund_Change
butHintergrund_Click (#)
10
Hintergrundbild-Modus wechseln
modHintergrund_Change
butPasswort_Click
11
Passwort-eingabe
frmKeyboard.Tag<>""
11
Passwort-eingabe (Kontrolle)
Eingabe unterschiedlich
12
Request: Fehler (1)
Password:=""
Eingaben gleich : Password:=frmKeyboard Tag
frmKeyboard Tag="":=Password
13
frmKiosk Maske setzen
frmKiosk_Paint
Button_Click (2)
Button_Click (3)
butAbbruch_Click
14
Request: Ende(1), Abbruch(2)
Button_Click (1)
15
Änderungen prüfen
Änderungen vorgenommen
16
Request: Speichern(1), Nicht speichern(2), Abbruch(3)
Button_Click (2)
18
frmInit schließen
frmInit_Unload
1
frmInit nicht geladen
Button_Click (1)
17
Änderungen speichern
datKioskIni_Write
butStart_Click
keine Änderungen vorgenommen
Initialisierung
Ebene: 1.6
Stand vom: 12.Nov.1994
Projekt: Kiosk V1.0
Bearbeiter:



Startverz. wechseln

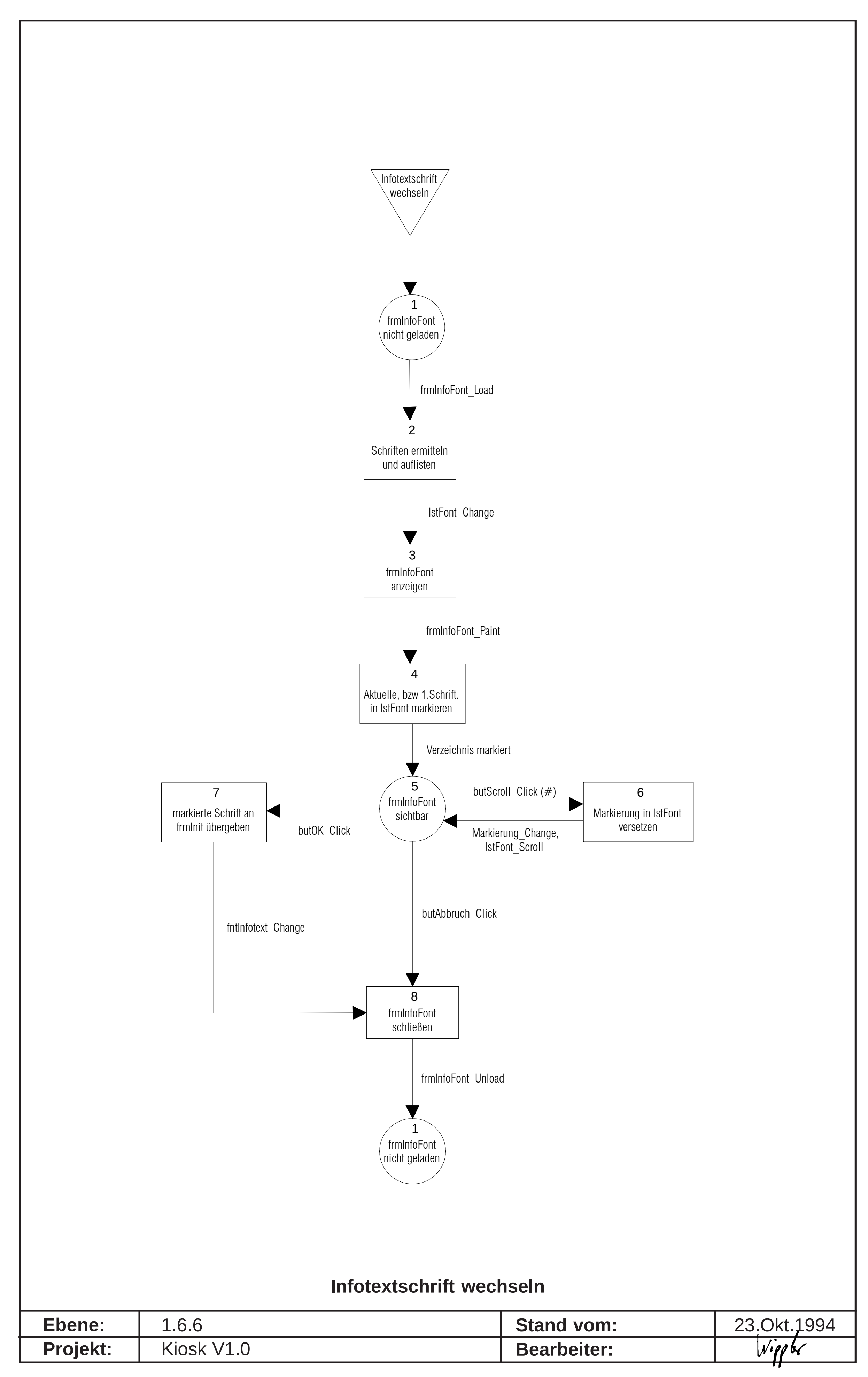
Ebene:	1.6.3	**Stand vom:**	23.Okt.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	


Buttonschrift wechseln
1 frmButtonFont nicht geladen
frmButtonFont_Load
2 Schriften ermitteln und in lstFont auflisten
lstFont_Change
3 Aktuelle, bzw 1.Schrift. in lstFont markieren
Schrift markiert
Schriftgröße_Change
4 Schriftgröße in txtFontSize anzeigen
txtFontSize_Change
7 Schriftgröße um 1 Pkt verkleinern
8 Schriftgröße um 1 Pkt vergrößern
5 Schriftattribute in txtFontDemo setzen
Markierung_Change, lstFont_Scroll
txtFontDemo_Change
butPlus_Click
6 frmButtonFont sichtbar
butMinus_Click
butScroll_Click (#)
9 Markierung in lstFont versetzen
butFett_Click
butOk_Click
12 markierte Schrift und Schriftgröße an frmInit übergeben
10 Fettschrift setzen
fntButtonBold_Change
butKursiv_Click
11 Kursicschrift setzen
fntButtonItalic_Change
butAbbruch_Click
neue Daten
13 frmButtonFont schließen
frmButtonFont_Unload
1 frmButtonFont nicht geladen
Buttonschrift wechseln
Ebene: 1.6.4
Stand vom: 23.Okt.1994
Projekt: Kiosk V1.0
Bearbeiter:



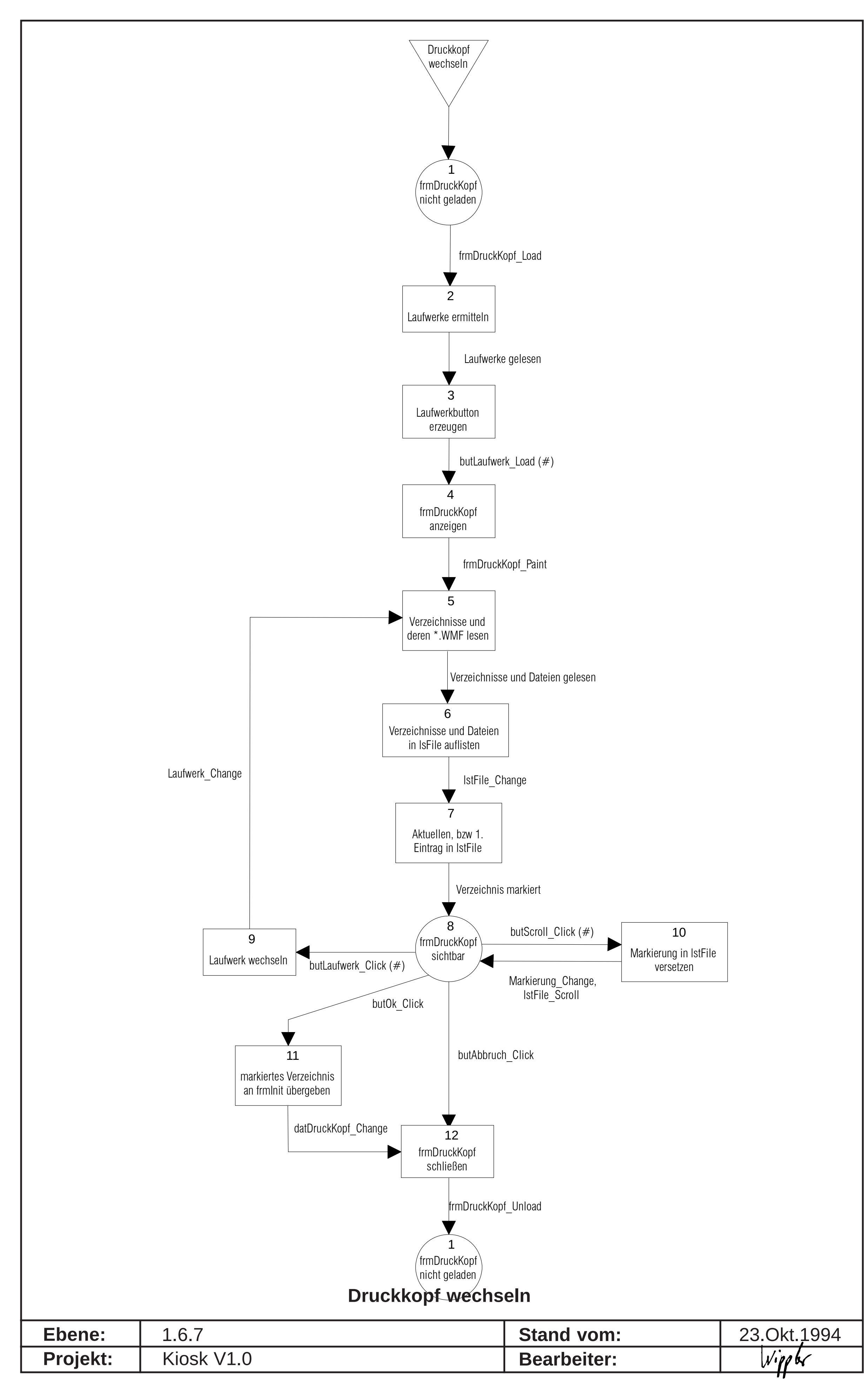
Farbe wechseln

Ebene:	1.6.5	**Stand vom:**	23.Okt.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	Wippler



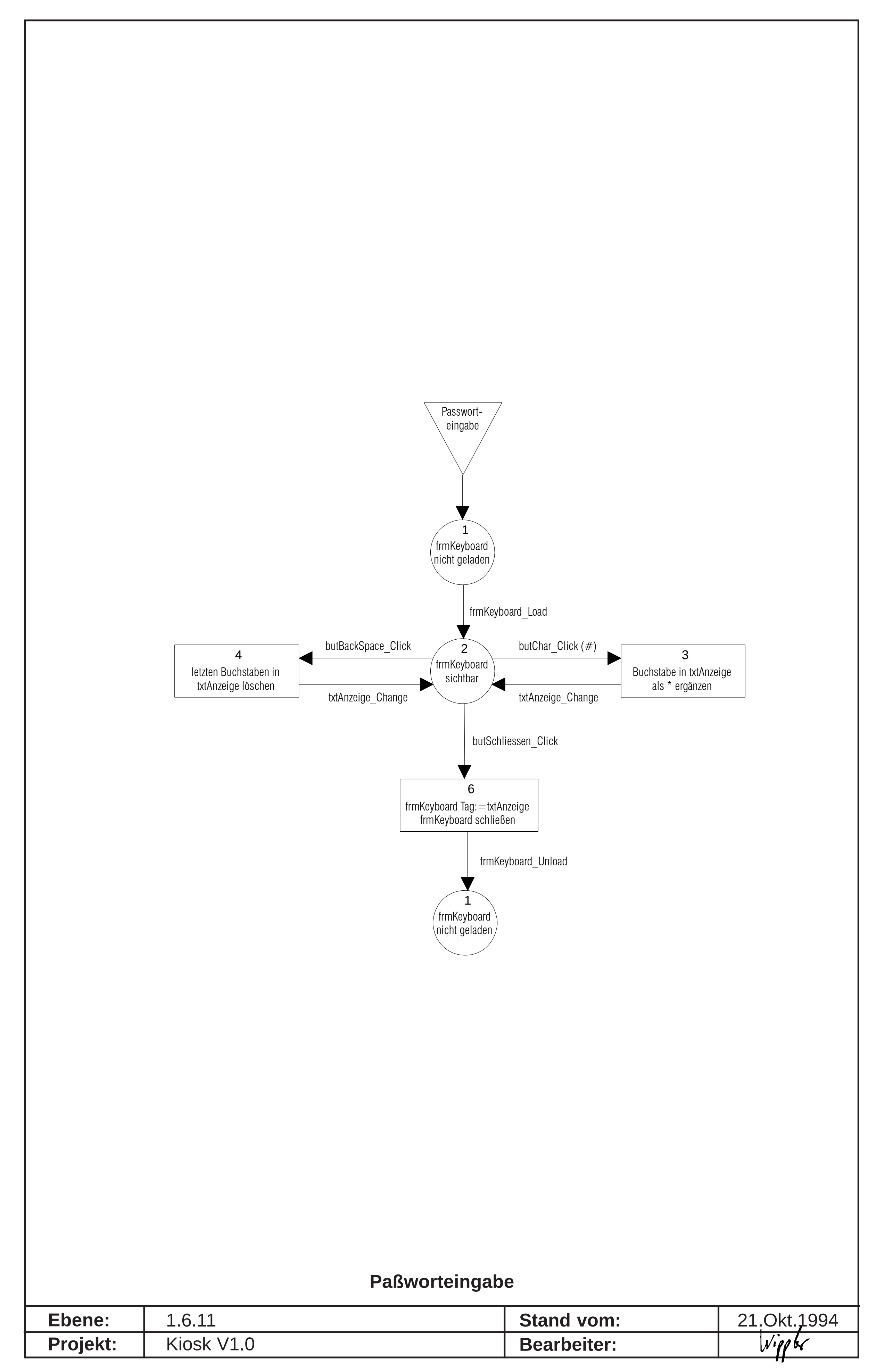
Infotextschrift wechseln

Ebene:	1.6.6	**Stand vom:**	23.Okt.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	Wipple



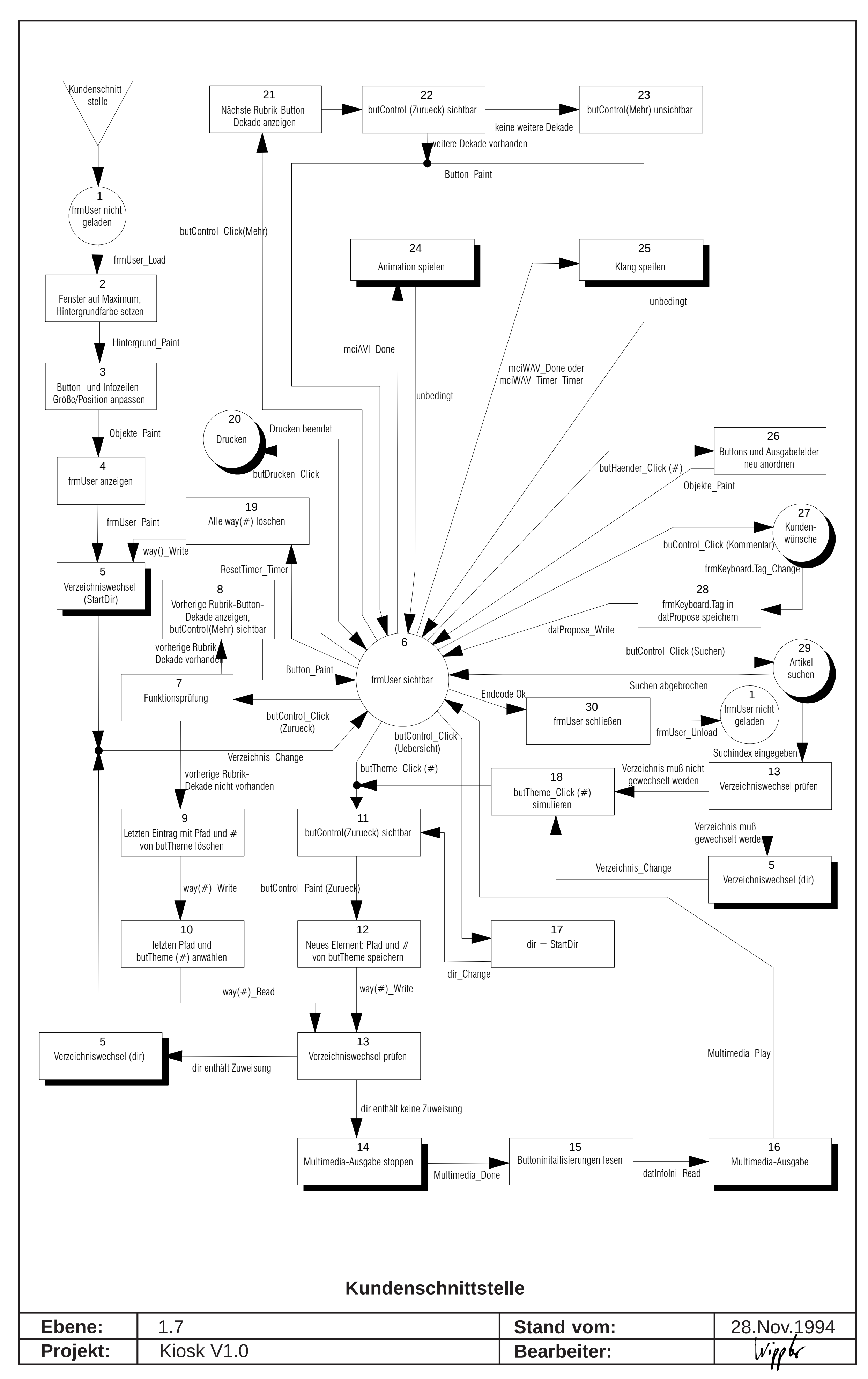
Druckkopf wechseln

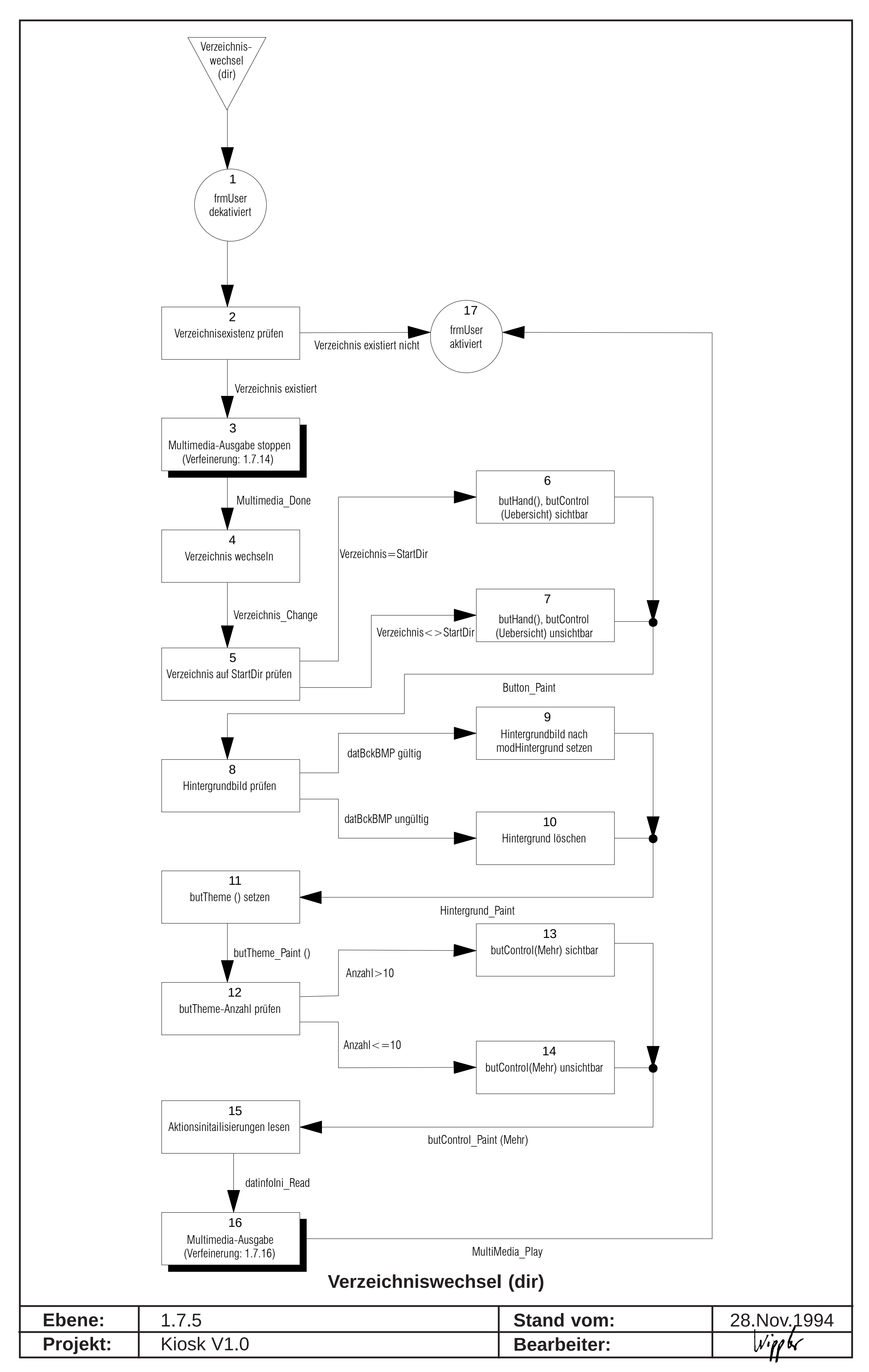
Ebene:	1.6.7	**Stand vom:**	23.Okt.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	[signature]



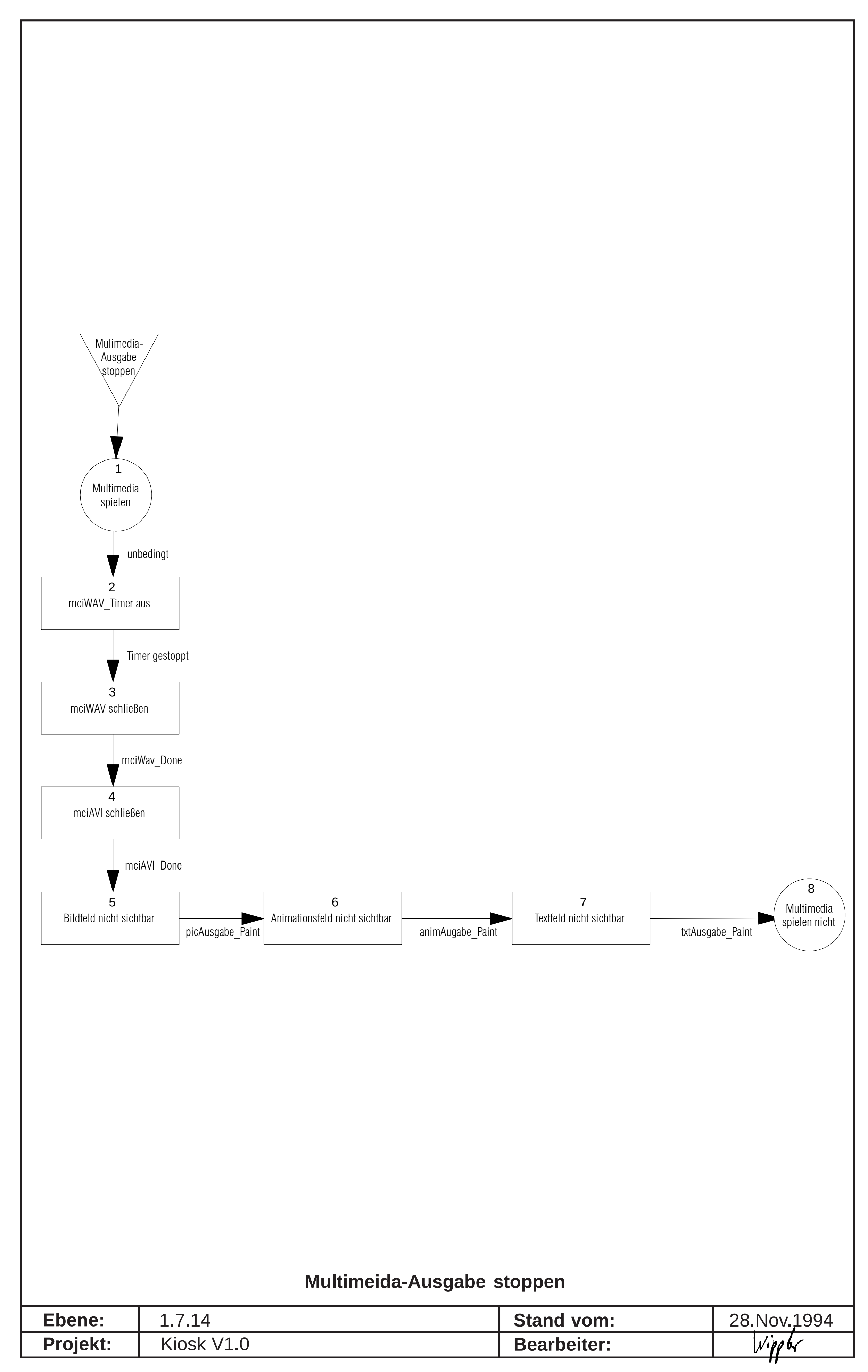
Paßworteingabe

Ebene:	1.6.11	**Stand vom:**	21.Okt.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	[illegible]

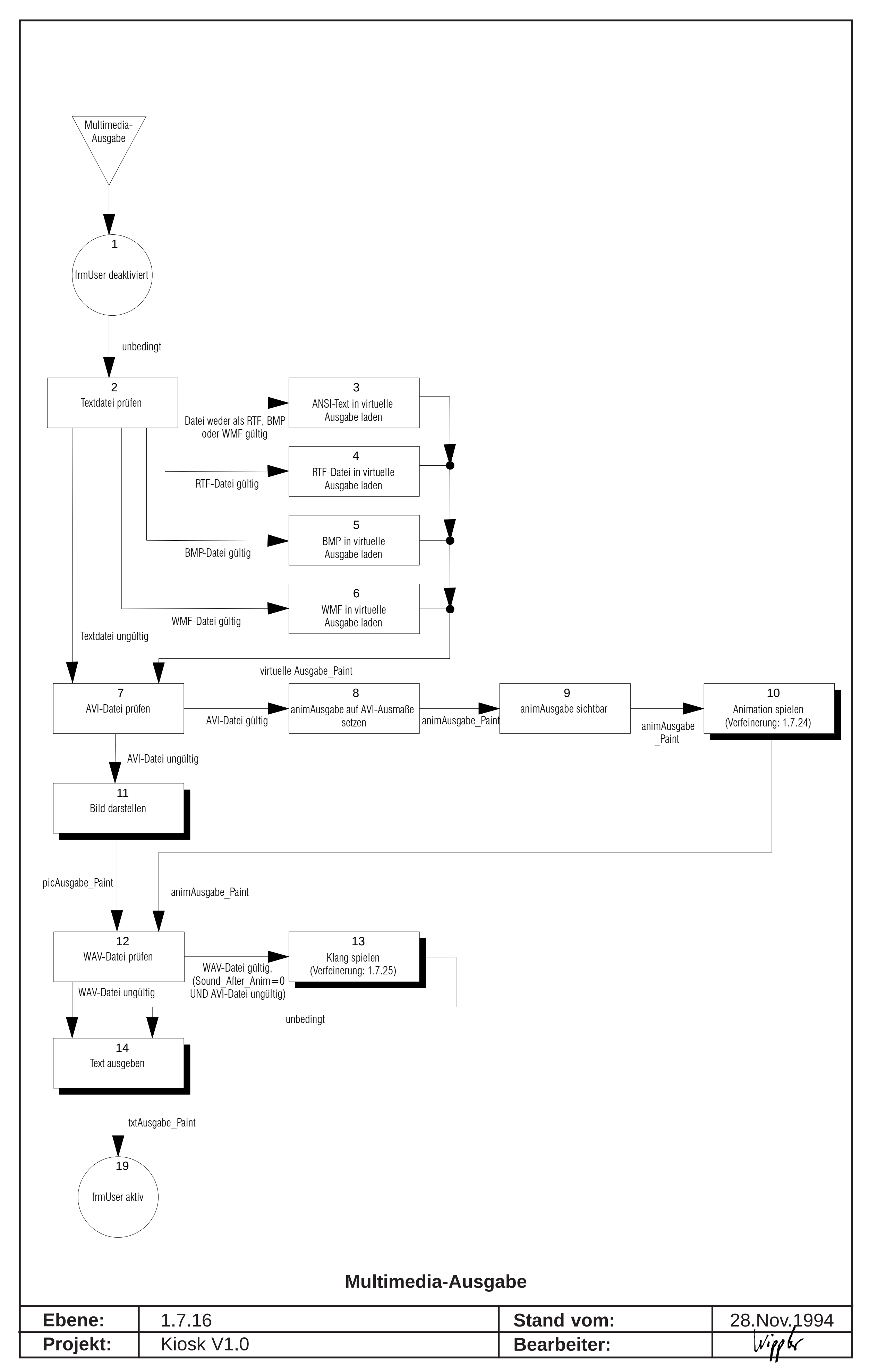

Kundenschnitt-
stelle
1
frmUser nicht
geladen
frmUser_Load
2
Fenster auf Maximum,
Hintergrundfarbe setzen
Hintergrund_Paint
3
Button- und Infozeilen-
Größe/Position anpassen
Objekte_Paint
4
frmUser anzeigen
frmUser_Paint
5
Verzeichniswechsel
(StartDir)
21
Nächste Rubrik-Button-
Dekade anzeigen
22
butControl (Zurueck) sichtbar
23
butControl(Mehr) unsichtbar
keine weitere Dekade
weitere Dekade vorhanden
Button_Paint
butControl_Click(Mehr)
24
Animation spielen
25
Klang spielen
unbedingt
mciAVI_Done
unbedingt
mciWAV_Done oder
mciWAV_Timer_Timer
20
Drucken
Drucken beendet
butDrucken_Click
19
Alle way(#) löschen
way()_Write
ResetTimer_Timer
8
Vorherige Rubrik-Button-
Dekade anzeigen,
butControl(Mehr) sichtbar
vorherige Rubrik-
Dekade vorhanden
Button_Paint
7
Funktionsprüfung
butControl_Click
(Zurueck)
6
frmUser sichtbar
26
Buttons und Ausgabefelder
neu anordnen
butHaender_Click (#)
Objekte_Paint
27
Kunden-
wünsche
buControl_Click (Kommentar)
frmKeyboard.Tag_Change
28
frmKeyboard.Tag in
datPropose speichern
datPropose_Write
29
Artikel
suchen
butControl_Click (Suchen)
Suchen abgebrochen
Endcode Ok
30
frmUser schließen
frmUser_Unload
1
frmUser nicht
geladen
Suchindex eingegeben
butControl_Click
(Uebersicht)
Verzeichnis_Change
butTheme_Click (#)
vorherige Rubrik-
Dekade nicht vorhanden
18
butTheme_Click (#)
simulieren
Verzeichnis muß nicht
gewechselt werden
13
Verzeichniswechsel prüfen
Verzeichnis muß
gewechselt werden
5
Verzeichniswechsel (dir)
Verzeichnis_Change
9
Letzten Eintrag mit Pfad und #
von butTheme löschen
11
butControl(Zurueck) sichtbar
way(#)_Write
butControl_Paint (Zurueck)
10
letzten Pfad und
butTheme (#) anwählen
12
Neues Element: Pfad und #
von butTheme speichern
17
dir = StartDir
dir_Change
way(#)_Read
way(#)_Write
5
Verzeichniswechsel (dir)
dir enthält Zuweisung
13
Verzeichniswechsel prüfen
Multimedia_Play
dir enthält keine Zuweisung
14
Multimedia-Ausgabe stoppen
Multimedia_Done
15
Buttoninitailisierungen lesen
datInfolni_Read
16
Multimedia-Ausgabe
Kundenschnittstelle
Ebene:
1.7
Stand vom:
28.Nov.1994
Projekt:
Kiosk V1.0
Bearbeiter:

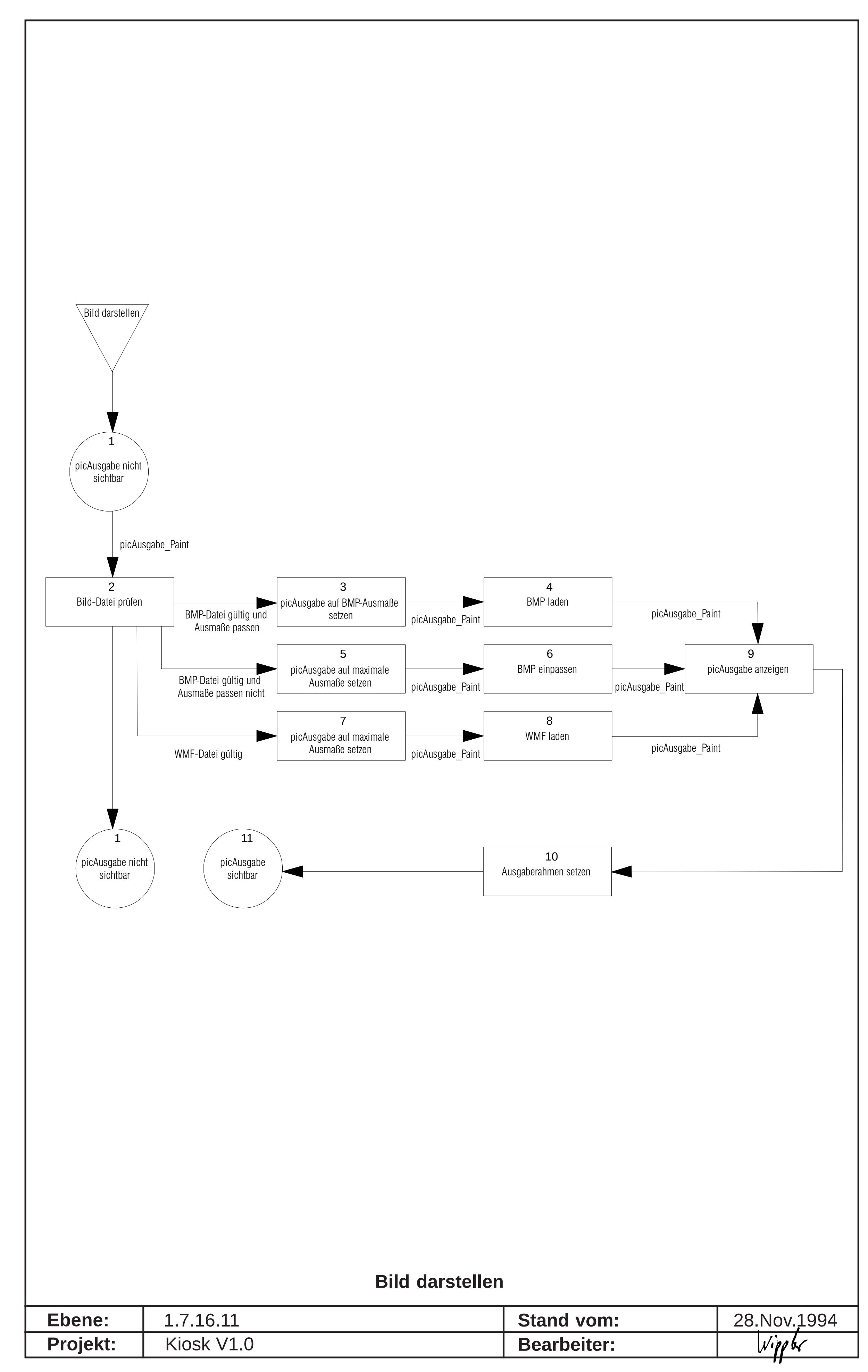


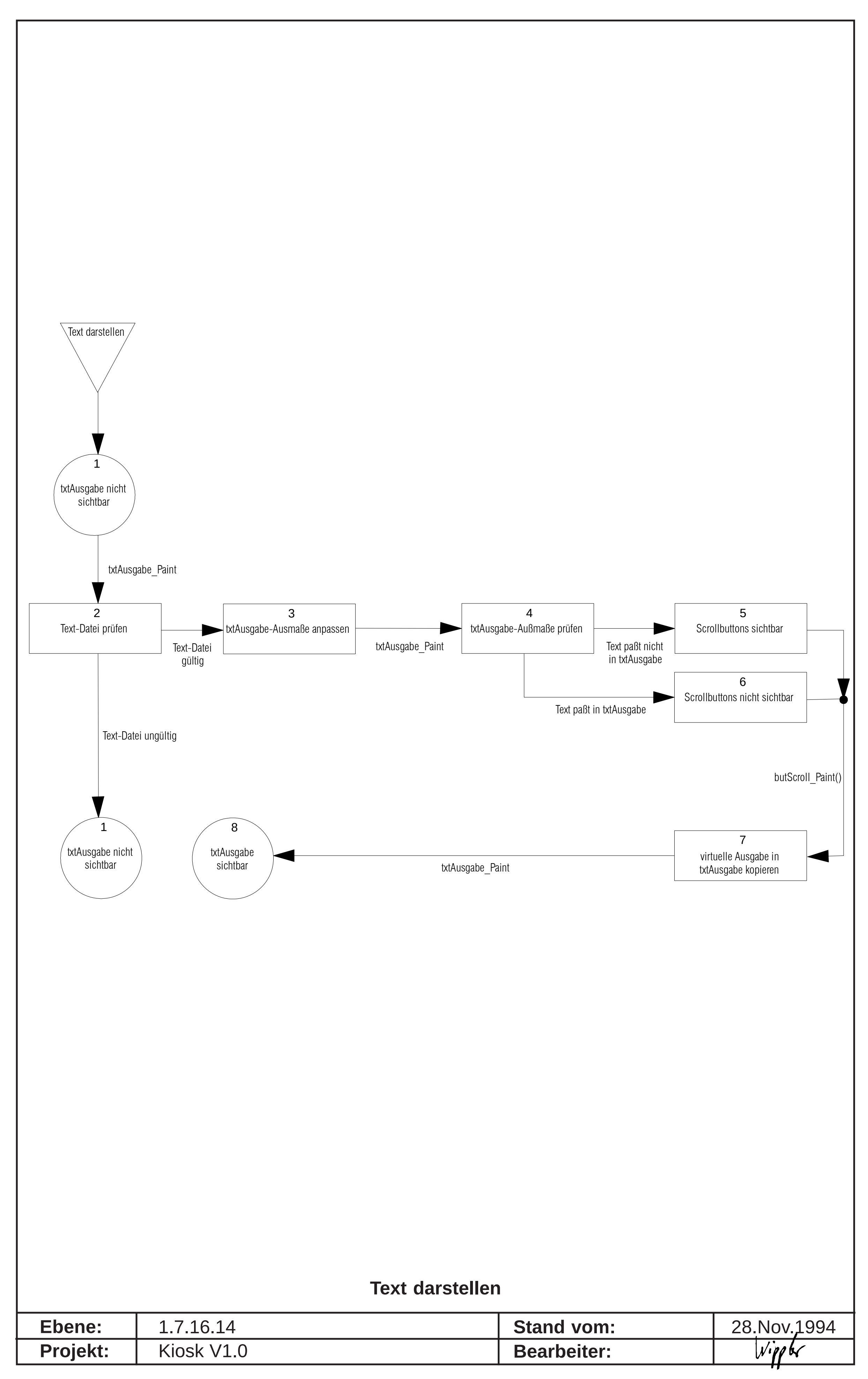
Verzeichnis-wechsel (dir)
1 frmUser dekativiert
2 Verzeichnisexistenz prüfen
Verzeichnis existiert nicht
17 frmUser aktiviert
Verzeichnis existiert
3 Multimedia-Ausgabe stoppen (Verfeinerung: 1.7.14)
Multimedia_Done
4 Verzeichnis wechseln
Verzeichnis_Change
5 Verzeichnis auf StartDir prüfen
Verzeichnis=StartDir
6 butHand(), butControl (Uebersicht) sichtbar
Verzeichnis<>StartDir
7 butHand(), butControl (Uebersicht) unsichtbar
Button_Paint
8 Hintergrundbild prüfen
datBckBMP gültig
9 Hintergrundbild nach modHintergrund setzen
datBckBMP ungültig
10 Hintergrund löschen
Hintergrund_Paint
11 butTheme () setzen
butTheme_Paint ()
12 butTheme-Anzahl prüfen
Anzahl>10
13 butControl(Mehr) sichtbar
Anzahl<=10
14 butControl(Mehr) unsichtbar
butControl_Paint (Mehr)
15 Aktionsinitailisierungen lesen
datinfoIni_Read
16 Multimedia-Ausgabe (Verfeinerung: 1.7.16)
MultiMedia_Play
Verzeichniswechsel (dir)
Ebene: 1.7.5
Stand vom: 28.Nov.1994
Projekt: Kiosk V1.0
Bearbeiter:



Ebene:	1.7.14	Stand vom:	28.Nov.1994
Projekt:	Kiosk V1.0	Bearbeiter:	

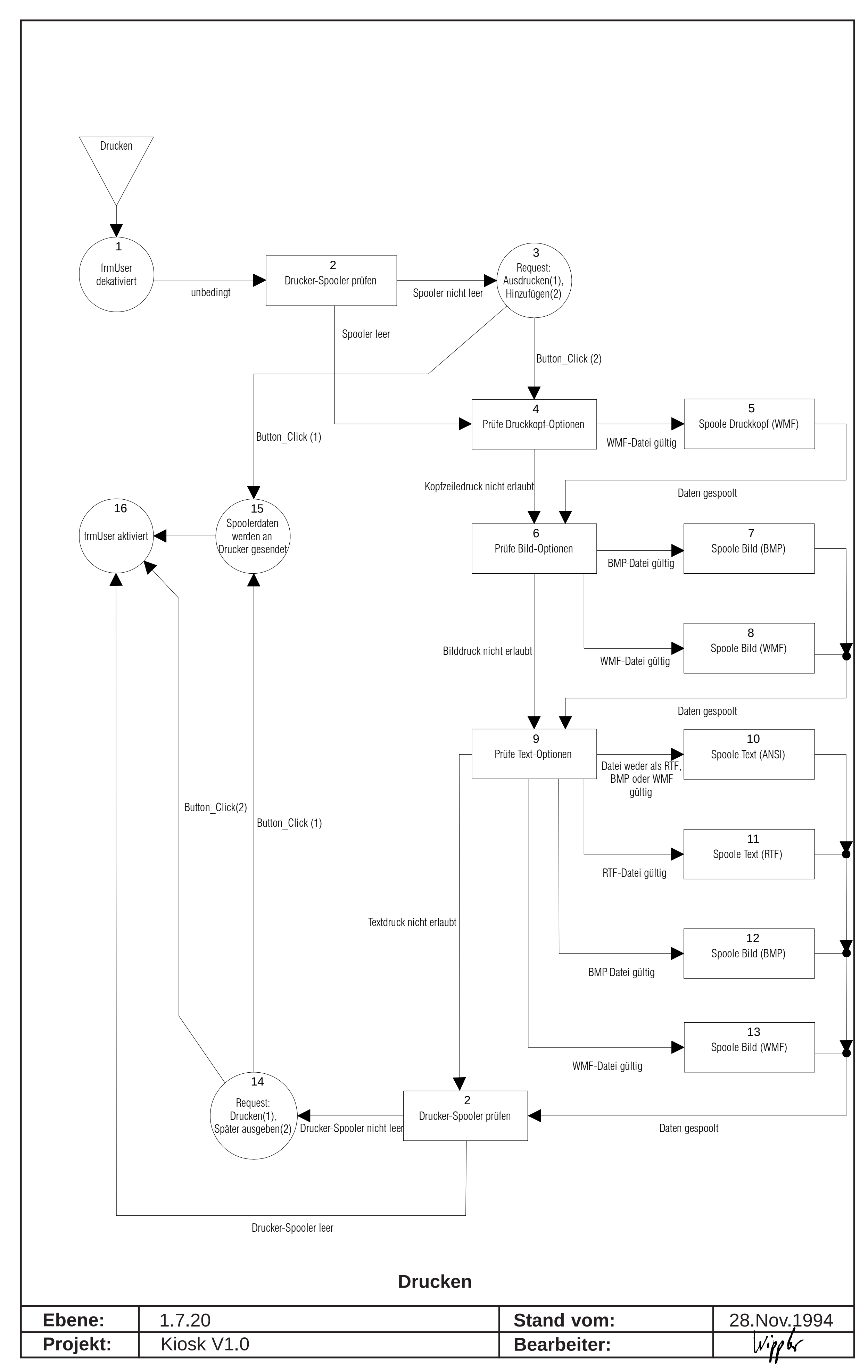

Multimedia-Ausgabe
1
frmUser deaktiviert
unbedingt
2
Textdatei prüfen
3
ANSI-Text in virtuelle Ausgabe laden
Datei weder als RTF, BMP oder WMF gültig
4
RTF-Datei in virtuelle Ausgabe laden
RTF-Datei gültig
5
BMP in virtuelle Ausgabe laden
BMP-Datei gültig
6
WMF in virtuelle Ausgabe laden
WMF-Datei gültig
Textdatei ungültig
virtuelle Ausgabe_Paint
7
AVI-Datei prüfen
AVI-Datei gültig
8
animAusgabe auf AVI-Ausmaße setzen
animAusgabe_Paint
9
animAusgabe sichtbar
animAusgabe _Paint
10
Animation spielen (Verfeinerung: 1.7.24)
AVI-Datei ungültig
11
Bild darstellen
picAusgabe_Paint
animAusgabe_Paint
12
WAV-Datei prüfen
13
Klang spielen (Verfeinerung: 1.7.25)
WAV-Datei gültig, (Sound_After_Anim=0 UND AVI-Datei ungültig)
WAV-Datei ungültig
unbedingt
14
Text ausgeben
txtAusgabe_Paint
19
frmUser aktiv
Multimedia-Ausgabe
Ebene: 1.7.16
Stand vom: 28.Nov.1994
Projekt: Kiosk V1.0
Bearbeiter:


Bild darstellen
1
picAusgabe nicht sichtbar
picAusgabe_Paint
2
Bild-Datei prüfen
BMP-Datei gültig und Ausmaße passen
3
picAusgabe auf BMP-Ausmaße setzen
picAusgabe_Paint
4
BMP laden
picAusgabe_Paint
BMP-Datei gültig und Ausmaße passen nicht
5
picAusgabe auf maximale Ausmaße setzen
picAusgabe_Paint
6
BMP einpassen
picAusgabe_Paint
9
picAusgabe anzeigen
WMF-Datei gültig
7
picAusgabe auf maximale Ausmaße setzen
picAusgabe_Paint
8
WMF laden
picAusgabe_Paint
1
picAusgabe nicht sichtbar
11
picAusgabe sichtbar
10
Ausgaberahmen setzen
Bild darstellen
Ebene:
1.7.16.11
Stand vom:
28.Nov.1994
Projekt:
Kiosk V1.0
Bearbeiter:



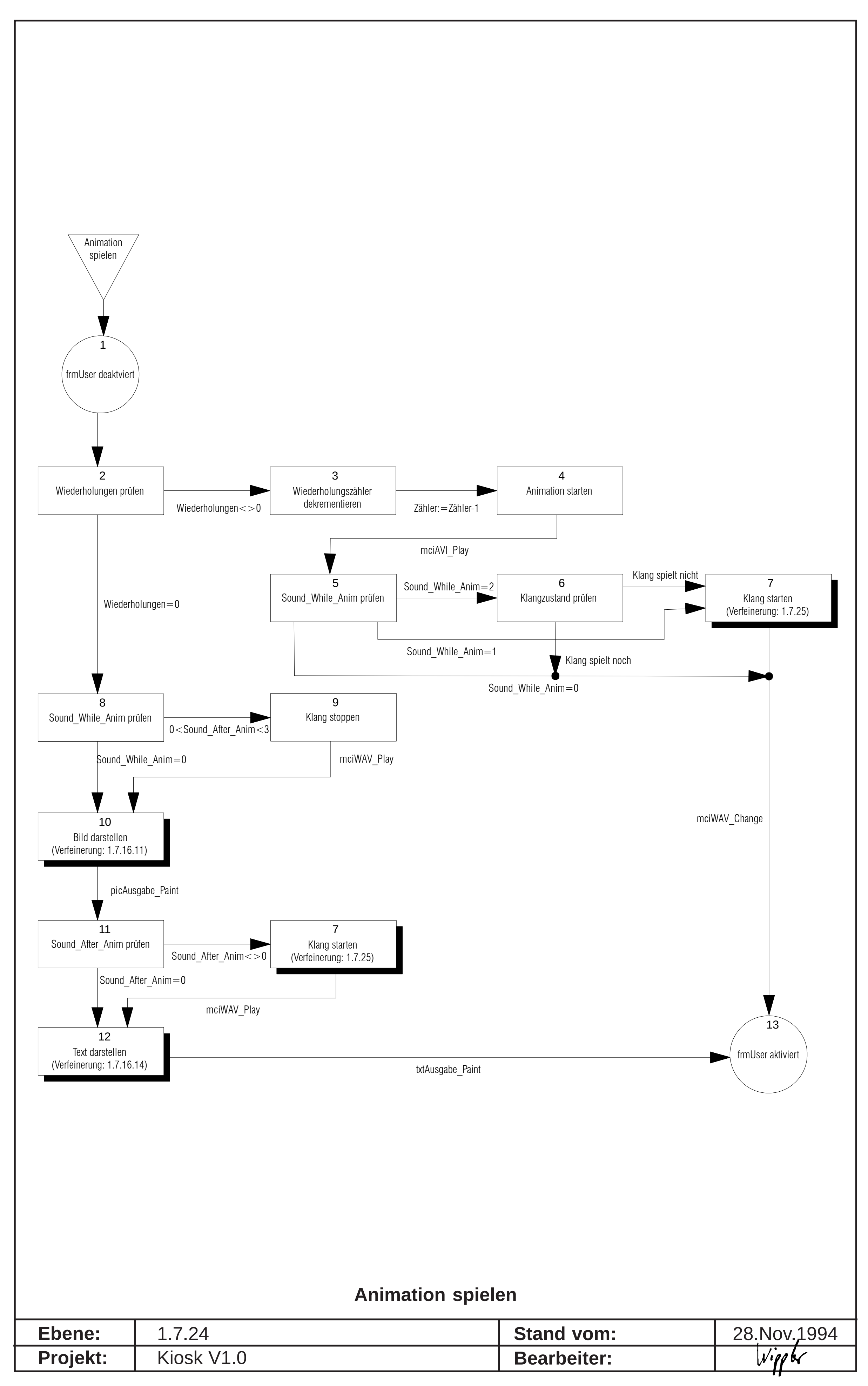
Text darstellen

Ebene:	1.7.16.14	**Stand vom:**	28.Nov.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	[signature]

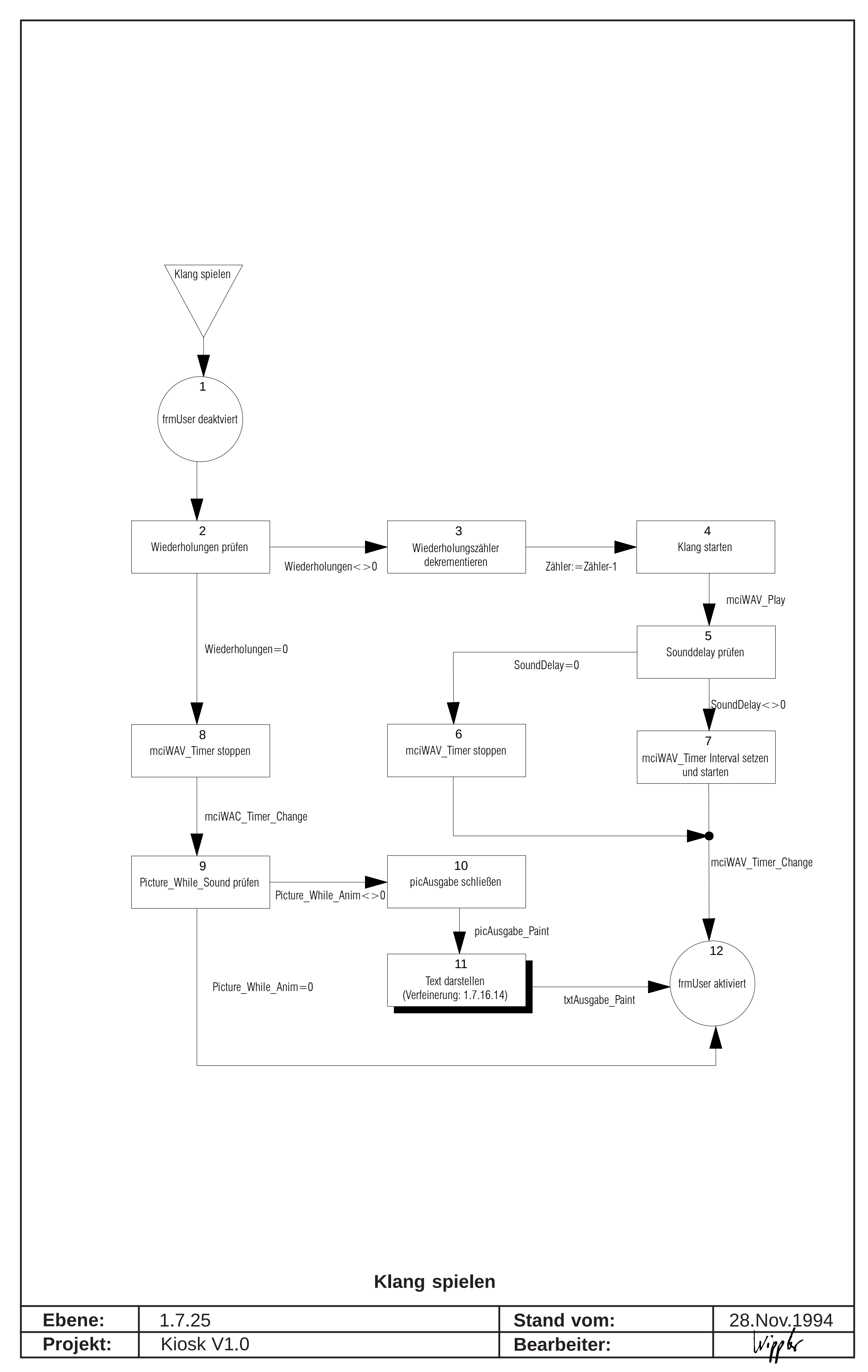


Drucken

Ebene:	1.7.20	Stand vom:	28.Nov.1994
Projekt:	Kiosk V1.0	Bearbeiter:	Wippler

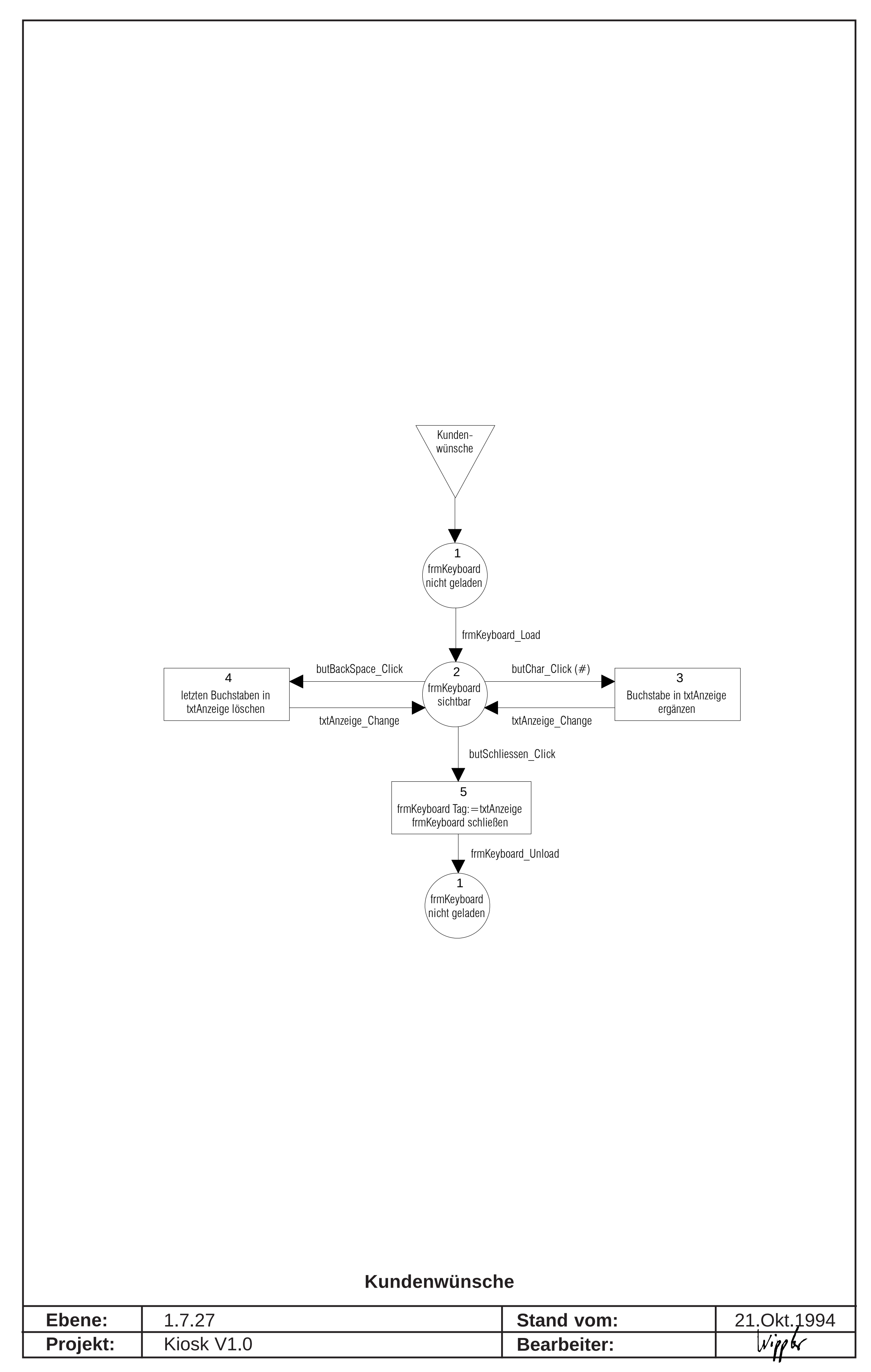


Animation spielen
1
frmUser deaktviert
2
Wiederholungen prüfen
Wiederholungen<>0
3
Wiederholungszähler dekrementieren
Zähler:=Zähler-1
4
Animation starten
mciAVI_Play
Wiederholungen=0
5
Sound_While_Anim prüfen
Sound_While_Anim=2
6
Klangzustand prüfen
Klang spielt nicht
7
Klang starten
(Verfeinerung: 1.7.25)
Sound_While_Anim=1
Klang spielt noch
Sound_While_Anim=0
8
Sound_While_Anim prüfen
0<Sound_After_Anim<3
9
Klang stoppen
Sound_While_Anim=0
mciWAV_Play
mciWAV_Change
10
Bild darstellen
(Verfeinerung: 1.7.16.11)
picAusgabe_Paint
11
Sound_After_Anim prüfen
Sound_After_Anim<>0
7
Klang starten
(Verfeinerung: 1.7.25)
Sound_After_Anim=0
mciWAV_Play
12
Text darstellen
(Verfeinerung: 1.7.16.14)
txtAusgabe_Paint
13
frmUser aktiviert
Animation spielen
Ebene:
1.7.24
Stand vom:
28.Nov.1994
Projekt:
Kiosk V1.0
Bearbeiter:



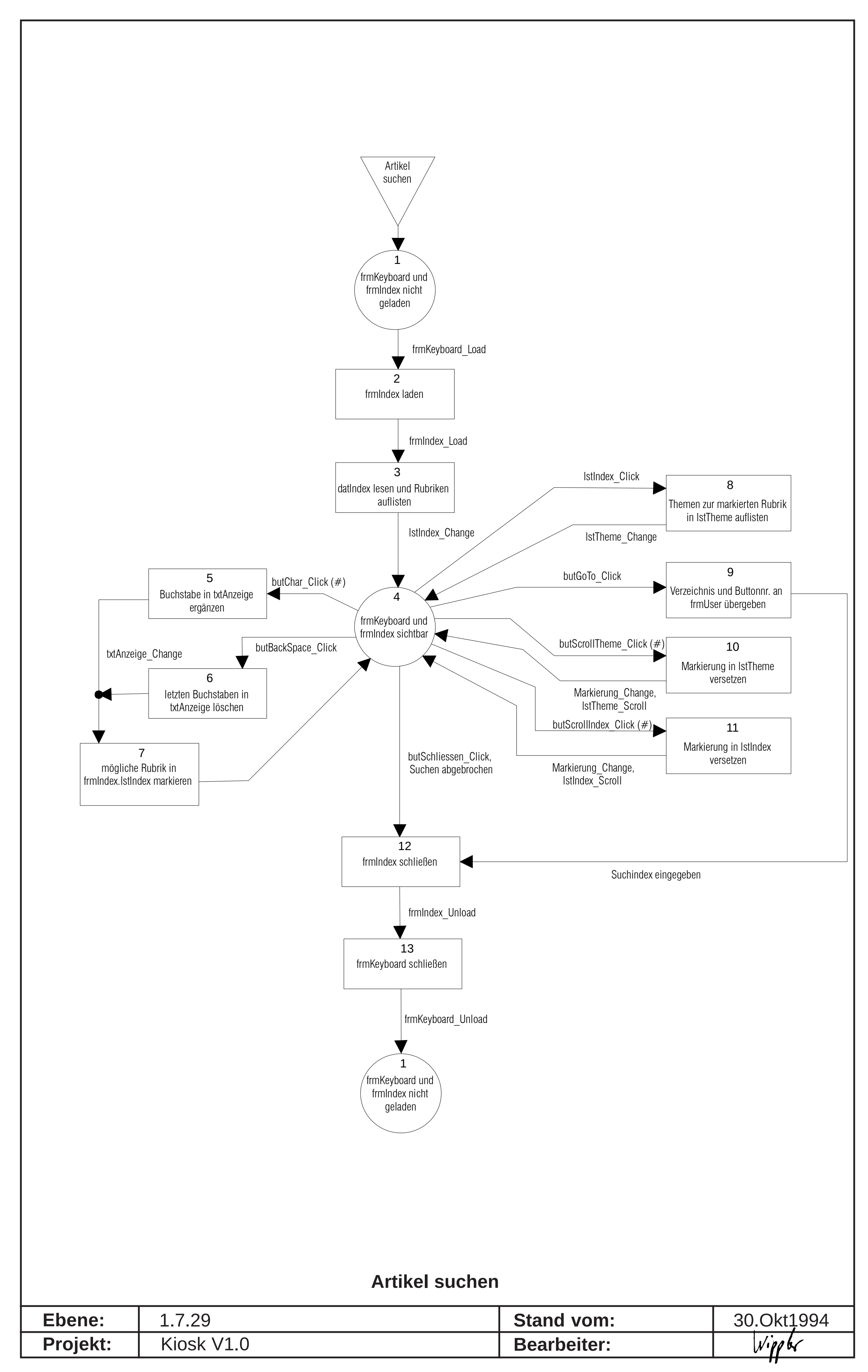
Klang spielen

Ebene:	1.7.25	**Stand vom:**	28.Nov.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	Wippler



Kundenwünsche

Ebene:	1.7.27	**Stand vom:**	21.Okt.1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	Wippler



Artikel suchen

Ebene:	1.7.29	**Stand vom:**	30.Okt1994
Projekt:	Kiosk V1.0	**Bearbeiter:**	Wippler



Software

4.1 Benutze Windows-Funktionen

Vom Programm werden einige nicht in Visual Basic 3.0 integrierte Windows-Funktionen verwendet. Eine Integration dieser Funktionen und Prozeduren erfolgen über den Befehl **Declare Sub** bzw. **Declare Function**, gefolgt vom Funktionsnamen, der Bibliothek und den Übergabeparametern. Ein Parameter wird ohne weitere Angaben als **Call-by-Referenz** übergeben. Soll ein **Call-by-Value** erfolgen, ist dies durch ein dem Parameter vorangehendes **ByVal** anzugeben.

GetPrivateProfileString (ByVal ApplicationName$, ByVal KeyName$, ByVal Default$, ByVal ReturnedString$, ByVal Size%, ByVal Filename$) as Integer

Die Funktion GetPrivateProfileString (LIB KERNEL) ermittelt eine Zeichenkette aus dem angegebenen Bereich innerhalb der angegebenen Initialisierungsdatei.

Parameterbeschreibung

ApplicationName zeigt auf einen Null-terminierten String, der den Bereich angibt, in welcher sich der Eintrag befindet.

KeyName zeigt auf den Null-terminierten String, der den Eintrag enthält, dessen zugehöriger String ermittelt werden soll.

Default zeigt auf einen Null-terminierten String, der den Standardwert des gewünschten Eintrags bestimmt, falls dieser Eintrag in der Initialisierungsdatei nicht gefunden werden kann. Dieser Parameter darf nicht “”sein.

ReturnedString zeigt auf den Puffer, in dem die Zeichenkette abgelegt werden soll.

Size gibt die Größe (in Bytes) des im Parameter *ReturnedString* angegebenen Puffers an.

Filename zeigt auf einen Null-terminierten String, der die Initialisierungsdatei benennt. Enthält dieser Parameter keine vollständige Pfadangabe, sucht Windows danach im Windows-Verzeichnis.

Rückgabewert

Der Rückgabewert gibt die Anzahl der in den angegebenen Puffer kopierten Bytes an. Die abschließenden Nullzeichen werden dabei nicht berücksichtigt.

function PrivateProfileString(ByVal ApplicationName$, ByVal KeyName$, ByVal Str$, ByVal FileName$) as Integer

Die Funktion WritePrivateProfileString (LIB KERNEL) kopiert einen Zeichen-String in den angegebenen Bereich einer Initialisierungsdatei.

Parameterbeschreibung

ApplicationName zeigt auf den Null-terminierten String, der den Bereich angibt, in den der String kopiert werden soll. Existiert dieser Bereich nicht, wird er neu angelegt. Der Name des Bereichs kann aus einer beliebigen Kombination von Groß- und Kleinbuchstaben bestehen. Die Schreibweise wird ignoriert.
KeyName zeigt auf einen Null-terminierten String, der den Namen des Eintrags enthält. der unter dem Programmkopf der Anwendung in der Initialisierungsdatei erscheint. Der Parameter *KeyName* ist “”, wenn der gesamte Bereich, der im Parameter *ApplicationName* angegeben wurde, zusammen mit allen Einträgen gelöscht werden soll.
Str zeigt auf den Null-terminierten String, der den neuen Wert des Eintrags enthält, oder ist “”, wenn der Eintrag, der im Parameter *KeyName* angegeben wurde, gelöscht werden soll.
Filename zeigt auf einen Null-terminiertenString, der die Initialisierungsdatei bezeichnet.

Rückgabewert

Der Rückgabewert ist ungleich 0, wenn die Funktion erfolgreich ausgeführt wurde. Andernfalls ist er 0.

function ShowCursor(ByVal Show%) as Integer

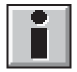
Die Funktion ShowCursor (LIB USER) zeigt oder versteckt den Mauszeiger.

Parameterbeschreibung

Show bestimmt, ob der Anzeigezählerstand erhöht oder verringert wird. Ist der Parameter TRUE, wird der Anzeigezählerstand erhöht. Andernfalls wird er verringert.

Rückgabewert

Der Rückgabewert gibt den neuen Anzeigezählerstand an, wenn die Funktion erfolgreich ausgeführt wurde.



Es gibt einen internen Anzeigezählerstand. Er steigt bei einem Show-Vorgang und sinkt bei einem Hide-Vorgang. Der Mauszeiger ist nur sichtbar, wenn der Anzeigezählerstand größer oder gleich Null ist.

procedure SetWindowPos(ByVal HWnd%,ByVal WndInsertAfter%, ByVal x%, ByVal y%, ByVal cx%, ByVal cy%, ByVal Flags%)

Die Routine SetWindowPos (LIB USER) ändert Größe, Position und Reihenfolge von untergeordneten Fenstern, Pop-Up- und Hauptfenstern. Fenster werden in der Reihenfolge ihres Erscheinens am Bildschirm angeordnet Das oberste Fenster erhält den höchsten Rang und ist das erste Fenster in der Liste.

Parameterbeschreibung

HWnd bestimmt das Fenster, das positioniert werden soll.
WndInsertAfter bezeichnet das Fenster in der Liste des Fenster-Managers, das dem positionierten Fenster in Z-Richtung vorangehen soll. Dieser Parameter setzt bei einem Wert = -1 das Fesnter an die oberste Stelle.
Die *x, y, cx, cy* positionieren das Fenster und bringen mit einem Wert = 0 keine Änderungen mit sich. Dieser Fall wird ausschließlich im Kiosk-System genutzt. *Flags* gibt Positionsoptionen des Fenster an. Im Programm wird ausschließlich die Konstante **SWP_NOACTIVATE = 40h** verwendet. Das Fenster wird an die oberste Stelle gesetzt und aktiviert.
Die Funktion wird benötigt, um die Tastatur in den Vordergrund zu laden, und gleichzietig das Index-Such-Fenster zur Benutzung freizugeben.

function GetTickCount as LongInt

Die Funktion GetTickCount (LIB USER) ermittelt die Anzahl von Millisekunden, die vergangen sind, seit Windows gestartet wurde.

Rückgabewert

Der Rückgabewert gibt die Anzahl von Millisekunden an, die vergangen sind, seit Windows gestartet wurde.

Diese Funktion wird für Verzögerungsroutine **Delay** benötigt.

Die nachfolgenden Routinen wurden im Pascal-Quellcode verwandt und sind nach den Konventionen von Borland Pascal 7.0 dargestellt.

```
function StretchDIBits(DC: HDC; DestX, DestY, DestWidth, DestHeight, SrcX, SrcY, SrcWidth, SrcHeight: Word;
        Bits: Pointer; var BitsInfo: TBitmapInfo;Usage: Word; Rop: LongInt): Integer;
```

Die Funktion StretchDIBits (Unit WinProcs) bewegt ein geräteunbhängiges Bitmap (*device-independent bitmap*, DIB) aus einem Quellrechteck in ein Zielrechteck und streckt oder staucht das Bitmap, falls erforderlich, um es den Abmessungen des Zielrechtecks anzupassen.

Parameterbeschreibung

DC legt den Zielgerätekontext für eine Bildschirmoberfläche oder ein Speicher-Bitmap fest.
DestX bestimmt die virtuelle x-Koordinate des Zielrechtecks.
DestY bestimmt die virtuelle y-Koordinate des Zielrechtecks.
DestWidth bestimmt die virtuelle x-Ausdehnung des Zielrechtecks.
DestHeight bestimmt die virtuelle y-Ausdehnung des Zielrechtecks.
SrcX bestimmt die x-Koordinate (in Pixel) des Quell-Rechtecks.
SrcY bestimmt die Koordinaten (in Pixel) des Quell-Rechtecks.im DIB.
SrcWidth bestimmt die Breite (in Pixel) des Quellrechtecks im DIB.
SrcHeight bestimmt die Höhe (in Pixel) des Quellrechtecks im DIB.
Bits zeigt auf die Bits des DIB, die in einem Array von Bytes abgelegt sind.
BitsInfo zeigt auf eine TBITMAPINFO-Datenstruktur, die Information über das DIB enthält.
Usage gibt an, ob die Elemente bmiColors[] des Parameters BitsInfo ausdrückliche RGB-Werte oder Indizes auf die gegenwärtig verwendete virtuelle Palette enthalten. Der Parameter *Usage* kann einer der folgenden Werte sein:
DIB_Pal_Colors, wenn die Farbtabelle aus einem Array von 16-Bit-Indizes auf die aktuell verwendete virtuelle

Palette besteht, oder **DIB_RGB_Colors**, wenn die Farbtabelle RGB-Werte enthält.

Wertbedeutung

DIB_Pal_Colors: Die Farbtabelle besteht aus einem Array von 16-Bit-Indizes auf die aktuell verwendete virtuelle Platte.
DIB_RGB_Colors: Die Farbtabelle enthält literale RGB Werte

Rop legt die vom GDI durchzuführende Rasteroperation fest. Die Codes der Rasteroperationen bestimmen, wie das GDI Farben bei Ausgabeoperationen kombiniert, die einen aktiven Pinsel, ein mögliches Quell-Bitmap und ein Zielbitmap einschließen. Die Liste der Raster-Operationscodes finden Sie unter BitBlt.

Rückgabewert

Der Rückgabewert ist die Anzahl der kopierten Scan-Zeilen, wenn die Funktion erfolgreich ausgeführt wurde.
StretchDIBits verwendet den Streckmodus des Zielgerätekontextes (gesetzt durch die Funktion SetStretchBltMode), um zu bestimmen, wie das Bitmap zu strecken oder zu stauchen ist.
Hier wurde der Modus **STRETCH_DELETESCAN** verwendet, der keine Farbveränderung beim Stauchen vornimmt.
Die Funktion wird zum Einpassen von BMP-Dateien benutzt.

function BitBlt(DestDC: HDC; X, Y, nWidth, Height: Integer; SrcDC: HDC; XSrc, YSrc: Integer; Rop: LongInt): Bool;

Die Funktion BitBlt (Unit WinProcs) kopiert ein Bitmap von einem angegebenen Gerätekontext auf einen Zielgerätekontext.

Parameterbeschreibung

DestDC bezeichnet den Gerätekontext, der das Bitmap aufnehmen soll.
X bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Zielrechtecks.
Y bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Zielrechtecks.
nWidth bestimmt (in virtuellen Einheiten) die Breite des Zielrechtecks und des Quell-Bitmaps.
Height bestimmt (in virtuellen Einheiten) die Höhe des Zielrechtecks und des Quell-Bitmaps.

SrcDC bezeichnet den Gerätekontext, aus dem das Bitmap kopiert wird. Muß den Wert Null haben, wenn der Parameter Rop eine Rasteroperation bezeichnet, die keine Quelle einschließt.
XSrc bestimmt die virtuelle x-Koordinate der oberen linken Ecke des Quell-Bitmaps.
YSrc bestimmt die virtuelle y-Koordinate der oberen linken Ecke des Quell-Bitmaps.
Rop bezeichnet die durchzuführende Rasteroperation. Der Wert bestimmt, wie die grafische Geräteschnittstelle (GDI) Farben in Ausgabevorgängen (wie ein aktuelles Spray, ein mögliches Quell-Bitmap sowie ein Ziel-Bitmap) kombiniert.
Dieser Parameter kann einen der folgenden Werte annehmen:

Codebeschreibung

SCRCOPY kopiert das Quell-Bitmap auf das Ziel-Bitmap. Da dies der einzig verwendete Parameter ist, wird auf weitere nicht eingegangen.

alle Angaben s.a. *Borland Pascal Hilfe 7.0*

4.2 Kommunikation Basic - Pascal

Aufgrund der unterschiedlichen Leistungsmerkmale der Programmiersprachen Visual Basic 3.0 und Borland Pascal 7.0, wurde eine Kombination beider gewählt. Der Funktionsaufruf von Pascal-Routinen aus Basic erfolgt über eine in Pascal compilierte DLL. Die Parameterübergabe kann sowohl als *Call-By-Value*, als auch *Call-By-Referenz* erfolgen. Im 1. Fall muß in Basic ein Parameter durch ein vorangehendes ***ByVal*** gekennzeichnet sein. Für den zweiten Fall ist in der Pascal-Routine ein ***Var*** voranzustellen. Für in Pascal angegebenen Typen werden in Basic folgende deklariert:

Pascal	Visual Basic
ShortInt, Integer, Byte	ByVal As Integer
THandle	ByVal As Integer
Word, LongInt	ByVal As Long
Real, Single	ByVal As Single
Double, Extended	ByVal As Double
PChar	ByVal As String
Record	ByVal As *Verbundvariable*

Vgl. *Das große Buch zu VisualBasic 3.0, 1993, S.598*



Benutzer-handbuch

5.1 Einleitung

Mit dem KIOSK V1.0 für Windows steht Ihnen ein umfangreiches Programm für die Präsentation und Wiedergabe verschiedenster Informationen zur Verfügung. Von Ihnen erstellte Grafiken, Animationen, Texte und Töne (Musik, Klang, Sprache) werden vom System über leicht zu erstellende Textdateien (*Makros*) integriert und bei einer Auswahl durch eine zweite Person wiedergegeben. Die Darstellung dieser ***Multi Media*** kann einzeln sowie gleichzeitig erfolgen, so daß der Informationsgehalt im Vergleich zum herkömmlichen Lesen von Prospekten erheblich verstärkt wird. Die Bedienung des Kiosk-Systems durch den Informationsempfänger erfolgt auf sehr leichte Weise. Die Software ist so ausgelegt, daß eine Steuerung über einen *Touchscreen* möglich ist. Schaltflächen werden nach Ihren Angaben in Makros vom Programm selbst erzeugt, so daß die Handhabung auch für den Anbieter sehr schnell zu erlernen ist. Die Makros beschränken sich auf ein notwendiges Minimum von Befehlen. Sie lassen sich mit jedem beliebigen Editor, der im ANSI-Format speichern kann, wie z.B. dem *Windows-Editor*, erstellen. Ein aufwendiges Erlernen des Umgangs mit Befehlen und einer umfangreichen Entwicklungsumgebung entfällt. Für die Anpassung der Software an Ihre Ansprüche und Ihre Hardware steht Ihnen eine leicht zu bedienende Initialisierungsmaske zur Verfügung.

5.2 Rechnerkonfiguration

Das Programm KIOSK V1.0 läuft auf jedem AT mit 4 MByte RAM, auf dem Windows 3.1 oder höher installiert ist. Jedoch empfiehlt sich, wegen der Größe von Animationen, Bildern und Klängen, eine Konfiguration mit mindestens 8 MByte. Da in vielen Fällen Grafiken und Animationen den Vorrang bei Präsentationen haben, sollte eine Grafikkarte installiert sein, die Echtfarbengrafiken, mindestens aber 64K-Farbgrafiken darstellen kann. Echtfarbenbilder werden bei einer niedrigeren Farbanzahl von Windows in den meisten Fällen sehr unansehnlich. Die Bildschirmauflösung ist beliebig, sollte jedoch so gewählt werden, daß Schriften gut lesbar bleiben. Hierzu reicht oft eine Auflösung von 640×480 Pixeln nicht aus. Als Kompromiß zwischen Darstellungsvielfalt, Geschwindigkeit und Preis wird ein 80486-DX2 Prozessor, 16 MByte Ram und eine Bildschirmauflösung von 800×600 Pixeln / 64K Farben empfohlen.

Die Größe der Festplatte hängt vom Umfang der Präsentationsdateien ab. Um in mehreren Stationen gleiche Daten zu integrieren, wird häufig das Medium CD-ROM genutzt, so daß ein entsprechendes Laufwerk und ggf. eine mindestens 640 MByte große Festplatte zur Aufnahme der Daten vorhanden sein sollte.

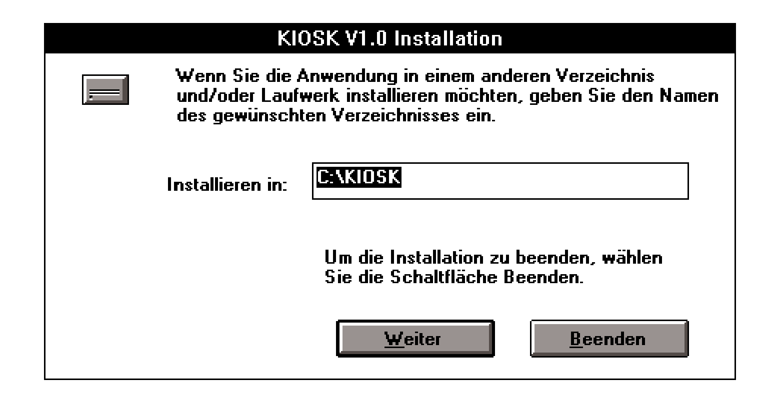
Für das Abspielen von Klängen muß eine Soundkarte, je nach Ansprüchen der Klangqualität, installiert sein. Im allgemeinen klingen 8-Bit-Samples zu dumpf, so daß eine 16-Bit-Soundkarte empfehlenswert ist.

Die Software ist so ausgelegt, daß die Tastatur überflüssig wird. Alle Eingaben erfolgen über die Maus, besser einem Trackball oder am besten über einen Touchscreen, der mittels Windows-Treiber die Maus ersetzt. Alle Schaltflächen sind ausreichend groß, so daß die Bedienung über einen 15’’-Monitor ohne Umstände erfolgen kann. Der Mauszeiger kann individuell ein- bzw. ausgeschaltet werden. Da Daten ggf. gedruckt werden können, sollte in diesem Fall ein Ihren Ansprüchen angepaßter Drucker angeschlossen und der entsprechende Treiber installiert sein.

5.3 Installation

Um Ihr System als Kiosk-System benutzen zu können, müssen Sie dieses Programm zunächst unter Windows installieren. Rufen Sie Windows auf und bleiben Sie im *Programm-Manager.* Wählen Sie aus dem Menü *Datei* den Eintrag *Ausführen*. Kurz darauf erscheint ein Dialogfeld, in dem Sie unter *Befehlszeile* je nach Laufwerkskonfiguration *A:\Setup* bzw. *B:\Setup* eingeben und mit *OK* bestätigen. Der Hinweis *Das Installationsprogramm wird initialisiert* ... teilt Ihnen mit, daß die Installation gestartet wurde und sich gleich auf Ihrem Bildschirm präsentieren wird.

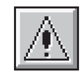
Der Installationsdialog



Es erscheint das Dialogfenster *KIOSK V1.0 Installation*. Geben Sie das Ziellaufwerk und Verzeichnis für das KIOSK-Programm ein. Nach der Betätigung von *Weiter* werden alle notwendigen Programmdaten in das angegebene bzw. in das Windows-Verzeichnis kopiert. Folgende Dateien werden dem Windows-Verzeichnis hinzugefügt:

Neue Dateien im Windows-Verzeichnis

- **MEDIA.DLL**
- **KIOSKDRV.DLL**
- **SYSTEM\VBRUN300.DLL**
- **SYSTEM\COMMDLG.DLL**
- **SYSTEM\MCI.VBX**
- **SYSTEM\THREED.VBX**
- **SYSTEM\CMDIALOG.VBX**
- **SYSTEM\MSOUTLIN**

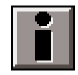


Sollte es bei der Installation der *commdlg.dll* zu Kollisionen mit anderen geöffneten Anwendungen kommen, wählen Sie im erscheinenden Dialogfeld *ignorieren*, da diese Datei dann bereits in Ihrem *WINDOWS\SYSTEM*-Verzeichnis installiert ist.

Sind alle Dateien kopiert, meldet sich das Kiosk-Installationsprogramm mit der Meldung:
Die Installation wurde erfolgreich abgeschlossen.
Die Programmgruppe *Kiosk* wurde im *Programm-Manager* automatisch hinzugefügt, aus der Sie die Anwendung durch einen Doppelklick starten können.

5.4 Benutzung durch den Anbieter

Begriffsdefinition



Für die weiteren Ausführungen gelten Sie als ***Anbieter*** und der Informationsempfänger als ***Kunde.***
Ein Teilbereich einer Präsentation wird nachfolgend als ***Thema*** bezeichnet.

5.4.1 Vorbereitung und Befehle

Richten Sie sich für die Präsentationsdateien ein Datenverzeichnis mit jeweiligen Unterverzeichnissen ein. Kopieren Sie dort die verschiedenen Präsentationsdateien hinein.

Dateiformate

Folgende Dateiformate können präsentiert werden:

Windows-Bitmaps

Bilder müssen im BMP-Format vorliegen und dürfen nur dann komprimiert sein (RLE4 oder RLE8), wenn die Bildschirmeinstellung mehr als 256 Farben zuläßt. Die Anzahl der Farben ist beliebig wählbar. Die Anzahl der Pixel ist ebenfalls beliebig, da ein zu großes Bild gestaucht wird. Berücksichtigen Sie, daß ein Bild nie breiter als 5/8 der Bildschirmbreite dargestellt wird (bei 800×600 Pixeln also 500 Pixel maximale Bildbreite). Größere Bilder kosten nur unnötige Umrechnungszeit. Jedes unter Windows lauffähige Bildbearbeitungsprogramm kann in der Regel in diesem Format speichern. Andere Formate werden von KIOSK V1.0 ignoriert und nicht dargestellt.

Windows-Metafiles

Vektorgrafiken müssen als WMF-Datei vorliegen und einen Header enthalten. Manche Programme, wie z.B. *CorelDraw!*, können beim Exportieren als Windows-Metafile diesen Header benutzerabhängig hinzufügen oder weglassen. Dateien ohne Header werden von KIOSK V1.0 ignoriert und nicht dargestellt.

Video For Windows

Animationen müssen im AVI-Format gespeichert werden. Die Komprimierungsart sowie die Farbanzahl ist beliebig. Jedoch sind sie in den Ausmaßen eingeschränkt. Animationen werden nur dann abgespielt, wenn ihre Breite × Höhe 5/8 × 1/2 der Bildschirmmaße nicht übersteigt (bei 800×600 max. 500×400 Pixel). Andere Formate, wie FLI oder FLC, müssen in AVI-Dateien konvertiert werden, da sie sonst ignoriert und nicht abgespielt werden.

Windows-Wave

Klangdateien müssen im WAV-Format gespeichert sein. Die Länge ist beliebig. Die Samplerate und Frequenz ist abhängig von den Möglichkeiten der installierten Soundkarte. Andere Formate werden ignoriert und nicht abgespielt.

MS Rich Text Format

Formatierte Texte können eingeschränkt wiedergegeben werden, wenn sie als RTF vorliegen, wie es z.B. *MS-Word für Windows* verarbeiten kann. Bilder im Text sowie Tabulatorstops, Spalten und Tabellen werden hierbei nicht berücksichtigt. Ebenso können Schriften nicht gemischt und unterschiedliche Formatierungen pro Absatz nur bei linksbündiger Ausgabe gesetzt werden. Andere Formate werden als ANSI-Text inkl. Steuerzeichen ausgegeben.

ANSI-Text

Jeder unformatierte Text kann wiedergegeben werden, wenn er im ANSI-Format vorliegt. Bei anderen Formaten ist nicht garantiert, daß alle Zeichen so wiedergegeben werden, wie es gewünscht wird.



Voraussetzung für die Wiedergabe von Klängen und Animationen ist die bereits durchgeführte Installation der entsprechenden Treiber unter der *Systemsteuerung*. Gehen Sie sicher, daß ***[MCI] Klang*** und ***[MCI] Microsoft Video for Windows*** installiert sind.

Initialisierung über Makros

In jedem von Ihnen eingerichteten Datenverzeichnis müssen die enthaltenen Dateien über eine ebenfalls von Ihnen zu erstellende Textdatei beschrieben werden. Über dieses sog. ***Makro*** wird KIOSK V1.0 während der Laufzeit initialisiert. Der Name eines Makros muß immer ***INFO.INI*** lauten. Diese Datei muß genau einmal in jedem Verzeichnis vorkommen. Sie muß im ANSI- bzw. ASCII-Format vorliegen und kann z.B. mit dem Windows-Editor bearbeitet und gespeichert werden. Definierte Themen, und somit darzustellende Schaltflächen (*engl. Buttons*), werden durch eckige Klammern begrenzt und beginnen immer mit dem Wort ***Button***, gefolgt von einem beliebigen, aber eindeutigen Namen.

Button definieren 

Beispiel:
[Button*Name*]

Name ist für die Indexerstellung notwendig, auf die noch eingegangen wird. In verschiedenen *INFO.INI* können Namen wiederholt werden. Die Groß- und Kleinschreibung wird hierbei nicht unterschieden.
Dieser Themenidentifizierung folgen nun die Makrobefehle. Diese definieren das Aussehen einer Schaltfläche und steuern den Ablauf der darzustellenden Dateien.

Makrobefehle

title=*beliebiger Text*
beschreibt den Text, der in den Schaltflächen wiedergegeben werden soll. *Beliebiger Text* ist die Überschrift des Themas.

thumbnail=*Pfad + Grafikdateiname*
weist der Schaltfläche ein kleines Bild zu. Das Bild kann als BMP oder WMF vorliegen und wird in die Schaltfläche eingepaßt. Die Zuweisung enthält den Pfad und den kompletten Namen der Bilddatei. Wird *Pfad* weggelassen, setzt KIOSK das aktuelle Verzeichnis voraus. Ist für *Pfad* **->** (Minus + größer als) angegeben, wird das unter **dir=...** beschriebene Verzeichnis eingesetzt. Soll in der Schaltfläche dasselbe wie unter **picture=...** zugewiesene Bild dargestellt werden, reicht die Zuweisung **thumbnail=->** aus.



Erzeugte Schaltfläche

Beispiel:
Die dargestellte Schaltfläche wird erzeugt über

```
[ButtonBeispiel:]
title=Sehen
thumbnail=augen.bmp
```

Wird eine Schaltfläche betätigt, kommen folgende Zuweisungen zur Wirkung:

dir=*Verzeichnisname*

Das System wechselt in das dem **dir** zugewiesene Verzeichnis. In diesem beschreibt eine weitere **INFO.INI** neue Schaltflächen mit anderen Eigenschaften. Für *Verzeichnisname* gelten die DOS-Bestimmungen. Es sind Angaben wie ***..\name , C:*** usw. gestattet. **dir** hat vor allen nachfolgenden Zuweisungen Vorrang. Es werden keine der Schaltfläche zugewiesenen Dateien dargestellt. Aktionen, die nach dem Öffnen eines Verzeichnisses ausgeführt werden, sind unter **[Action]** beschrieben.

Animationsoptionen

animation=*Pfad+AVI-Dateiname*

Ist eine gültige AVI-Datei zugewiesen, wird diese geladen, einmal abgespielt und wieder vom Bildschirm genommen.
Um eine Animation mehrmals abspielen zu lassen, dient der nächste Befehl.

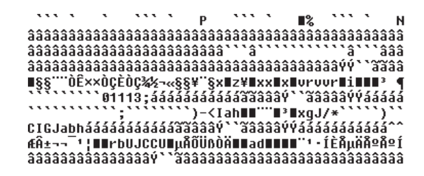
animreplay=*Anzahl*

Anzahl entspricht einer ganzen Zahl und gibt an, wie oft eine Animation abgespielt werden soll. Für eine Endloswiederholung geben Sie für *Anzahl* eine negative Zahl ein.

Bildoption

picture=*Pfad + Grafikdateiname*

Entspricht der zugewiesene Name einer BMP- bzw. WMF-Datei, wird diese in Original- bzw. gestauchter Größe dargestellt. Voraussetzung hierfür ist, daß alle Animationswiederholungen beendet sind. Läuft eine Animation, erfolgt die Bilddarstellung erst im Anschluß.
Mit Ausnahme der unter **Picture_While_Sound=1** beschriebenen Zuweisung bleibt das Bild bis zum nächsten Themenwechsel sichtbar.



Ausgabe einer TIFF-Datei

Texte im RTF-, TXT-, BMP- oder WMF-Format

text=*Pfad + Textdateiname*

Die hier zugewiesene Datei kann grundsätzlich ein beliebiges Format haben. Jedoch gibt z.B. eine Grafikdatei im TIF-Format nicht sehr viele Information wieder. Die Ausgabe von Bildern ist aber nicht unmöglich. Wenn *Textdateiname* eine Grafik im BMP- bzw. WMF-Format ist, wird diese in das Textfenster eingepaßt. Textdateien können ebenso im ANSI-Format oder MS Rich Text Format ausgegeben werden. Der Aufbau von RTF-Dateien wird später beschrieben. Bei RTF-Texten werden eingeschränkt Formatierungen für die Ausgabe umgesetzt. Bei der Textausgabe findet ein automatischer Zeilenumbruch statt. Texte werden zeitgleich mit Animationen bzw. Bildern und Klängen ausgegeben. Sollte ein Text oder Bild länger als der zur Verfügung stehende Platz sein, kann die Ausgabe über zwei Pfeiltasten im Sichtfenster verschoben werden.

Klangoptionen

sound=*Pfad + Klangdateiname*

Ist hier eine im WAVE-Format gespeicherte Datei zugewiesen, wird diese mit der Darstellung einer Animation, bzw. eines Bildes zeitgleich abgespielt. Dies geschieht genau einmal. Um einen Klang mehrmals abspielen zu lassen, dient der nächste Befehl.

soundreplay=*Anzahl*

Anzahl entspricht einer ganzen Zahl und gibt an, wie oft ein Klang abgespielt werden soll. Für eine Endloswiederholung geben Sie für *Anzahl* eine negative Zahl ein.

sounddelay=*Pause*

Pause gibt an, wieviel Sekunden auf eine Klangwiederholung gewartet werden soll. Maximal sind 60 Sekunden möglich.

Sound_After_Anim=*Option*

Mit *Option = 0* hat diese Zuweisung keine Wirkung. Mit *Option = 1* wird die initialisierte Klangdatei erst nach dem Abspielen einer Animation inkl. allen Wiederholungen, wiedergegeben.

Picture_While_Sound=*Option*

Mit *Option = 0* hat diese Zuweisung keine Wirkung.

Mit *Option = 1* wird ein geladenes Bild nur so lange gezeigt, bis ein Klang inkl. allen Wiederholungen abgespielt ist.

Sound_While_Anim=*Option*

Mit *Option = 0* hat diese Zuweisung keine Wirkung. Mit *Option = 1* beginnt der Klang, unabhängig von seiner Länge und der Verzögerung, bei jeder Animationswiederholung von vorn. Der Klang wird so oft wiedergegeben, wie es unter **soundreplay** zugewiesen ist.

Mit *Option = 2* wird die initialisierte Klangdatei so lange wiederholt, bis die Animation mit allen Wiederholungen abgespielt ist, oder alle Wiederholungen des Klanges ausgeführt sind. Der Klang wird immer komplett gespielt. Vor jeder Wiederholung erfolgt eine Pause der unter **sounddelay** angegebenen Sekunden.

Druckoption

print=*Option*

Ist *Option = 0* hat diese Zuweisung keine Wirkung. Über *Option = 1* wird dem Kunden gestattet, die initialisierten Grafiken und Texte auszudrucken.

Befehlshierarchie

Die Reihenfolge der Befehle ist beliebig wählbar. Die Verwendung eines Befehls ist nur bei einer gültigen Zuweisung notwendig. Falsch geschriebene Befehle werden ignoriert. Die Groß- und Kleinschreibung wird nicht beachtet. Die Buttons werden in der Reihenfolge dargestellt, wie sie in der INFO.INI von oben nach unten wiederzufinden sind.

Verzeichniswechsel

[Action]

Unterbereiche eines Themas werden dann dargestellt, wenn über **dir=...** ein anderes Verzeichnis geöffnet wird. Die dort enthaltene INFO.INI beschreibt neue Buttons, die entsprechend dargestellt werden. Zuzüglich zur Definition von Buttons können Aktionen direkt nach dem Öffnen des Verzeichnisses erfolgen, indem der Kennung **[Action]** die vorangehend beschriebenen Befehle folgen. Ausnahmen sind die Zuweisungen **title**, **thumbnail**, **dir** und **print**, die wegfallen können, da sie ggf. unbeachtet bleiben.

INFO.INI 

Beispiel:
Ein Verzeichnis enthält folgende Dateien:
INFO.INI,
COMPUTER.AVI, MMEDIA.AVI,
AUGEN.BMP, MUSIK.BMP, UHR.BMP,
SPRACHE.WAV, MUSIK.WAV, TICK.WAV,
AUGEN.RTF,
und das Unterverzeichnis PLANETEN mit der Datei ERDE.BMP

Inhalt der INFO.INI

```
[Action]
animation=computer.avi
animreplay=3

[buttonTime]
title=Zeit sparen
thumbnail=->
picture=uhr.bmp
sound=tick.wav
soundreplay=-1
sounddelay=1

[buttonMMedia]
title=Multimedia
animation=mmedia.avi
animreplay=-1

[buttonAugen]
title=Sehen
thumbnail=->
picture=augen.bmp
text=Augen.rtf
sound=sprache.wav

[buttonMusik]
title=Hören und Sehen
thumbnail=->
picture=musik.bmp
sound=hiphop.wav
soundreplay=-1
sounddelay=10
animation=mmedia.avi
animreplay=3

[ButtonNext]
title=Anderes Thema
thumbnail=->erde.bmp
dir=planeten
```

Die links dargestellte INFO.INI erzeugt nach dem Öffnen des Verzeichnisses folgende Buttons











und spielt die Animation COMPUTER.AVI dreimal ab.

Bei Betätigung von *Zeit sparen* wird das Bild der Uhr groß dargestellt, und jede Sekunde erfolgt ein Ticken. Bild und Ton sind so lange wahrnehmbar, bis ein weiterer Button betätigt wird.

Der Button *Multimedia* lädt die Animation MMEDIA.AVI und spielt sie so lange, bis ein weiterer Button betätigt wird.

Sehen lädt das Bild der im Button sichtbaren Augen, zeigt den Text AUGEN.RTF und spricht über SPRACHE.WAV.

Über *Hören und Sehen* wird die Animation MMEDIA.AVI zusammen mit dem sich alle zehn Sekunden wiederholenden Klang HIPHOP.WAV dreimal

abgespielt. Anschließend wird das Bild MUSIK.BMP gezeigt. Der Klang wird weiterhin alle zehn Sekunden wiederholt.

Anderes Thema erhält sein kleines Bild aus dem Verzeichnis PLANETEN, in der die Datei ERDE.BMP enthalten ist. Bei Betätigung wird in dieses Verzeichnis gewechselt und neue Buttons ggf. nach der dort gespeicherten INFO.INI gesetzt.

Indexerstellung

Dem Kunden kann die Möglichkeit geboten werden, über direkte Eingabe eines Bereiches und eines dazugehörenden Themas, sofort zum gewünschten Punkt zu kommen, ohne den müßigen Weg des Durchschaltens gehen zu müssen. Dazu muß von Ihnen eine Datei mit Namen ***INDEX.INI*** im ANSI-Format in Ihrem KIOSK-Verzeichnis gespeichert werden. Das KIOSK-Verzeichnis ist das Verzeichnis, in dem KIOSK.EXE installiert wurde. Dies entspricht nicht unbedingt dem Datenverzeichnis. Der Aufbau der *INDEX.INI* sieht wie folgt aus:
Alle Themen werden durch eine Gruppenbezeichnung zusammengefaßt. Dieser Gruppen- oder Bereichsname wird in eckigen Klammern über die zugehörigen Themen geschrieben.

INDEX.INI Gruppendefinition

Beispiel:

```
[Beispiele]
```

Darunter werden die Themen, deren Verzeichnis und die darin beschriebenen Schaltflächen in folgender Weise notiert:

Thema=Verzeichnis,Button

Thema ist die Bezeichnung, unter der in einer Auswahlliste dasselbe wiederzufinden ist.

Verzeichnis gibt den Pfad, ausgehend vom Datenverzeichnis an, in dem das Thema über die dort enthaltene *INFO.INI* beschrieben ist.

Button gibt den Namen der Schaltfläche an, der vom Programm betätigt werden soll, sobald das entsprechende Verzeichnis geöffnet wurde. Dieser Name muß dem unter *Initialisierung über Makros* beschriebenen Anhang der Bezeichnung **[Button*Anhang*]** entsprechen. Für die vorangehend beschriebene *INFO.INI* ist die nachfolgende *INDEX.INI* gespeichert worden.

INDEX.INI 

Inhalt der INDEX.INI

```
[Vorteile]
Zeit sparen=,Time

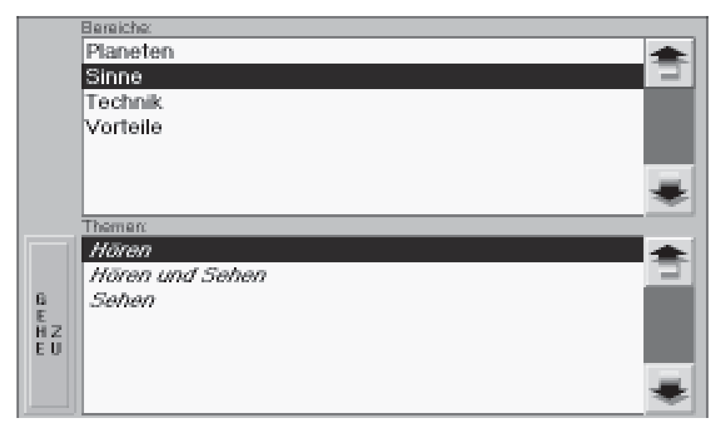
[Technik]
Multi Media=,MMedia

[Sinne]
Sehen=,Augen
Hören=,Musik
Hören und Sehen=,Musik

[Planeten]
Die Erde=planeten,next
```

Beispiel:
Im KIOSK-Verzeichnis ist die links beschriebene *INDEX.INI* enthalten.

Die in eckigen Klammern stehenden Bezeichnungen sind Themenbereiche oder auch Gruppennamen. Sie erscheinen für den Kunden in einer Bereichsliste, aus der er wählen kann. Jeweils einem Bereich werden Themen zugeordnet. Wird unter *Sinne* das Thema *Hören* gewählt, wechselt KIOSK nicht das Verzeichnis (leere Eingabe für *Pfad*) und führt die in der *INFO.INI* unter **[buttonMusik]** stehenden Befehle aus.
Wurde *Planeten-Die Erde* gewählt, wechselt KIOSK in das Verzeichnis *Planeten* und betätigt den *ButtonNext*.



Listen für die Themensuche über einen Index

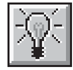
Pfade und Buttons, die nicht existieren, werden ignoriert. Wenn über *Verzeichnis* in ein Verzeichnis gewechselt werden soll, das kein Unterverzeichnis des Datenverzeichnisses ist, haben Sie zwar eine verworrene Verzeichnisstruktur zu verzeichnen, können aber durch Angabe des gesamten Pfades in dieses Verzeichnis wechseln. **Kurz:** Sie müssen Ihre Angaben mit der Laufwerkskennung oder einfach einem Backslash “\” beginnen, wenn das Thema nicht im Datenverzeichnis untergebracht ist.

Die Vorbereitung ist hiermit abgeschlossen. Sie müssen bis hier KIOSK V1.0 installiert und ein Datenverzeichnis erstellt haben, in dem sich Unterverzeichnisse mit diversen BMP-, WAV-, AVI-, RTF-, WMF-und TXT-Dateien und einer *INFO.INI* befinden. Ggf. enthält das KIOSK-Verzeichnis eine *INDEX.INI*. **Alle weiteren Eingaben können über einen Touchscreen erfolgen.**

5.4.2 Formatierte Textausgabe

Das MS Rich Text Format

Wie bereits erwähnt, besteht die Möglichkeit, formatierte Texte ausgeben zu lassen, sofern sie als RTF-Datei vorliegen. Solch eine Datei ist grundsätzlich mit jedem Editor bearbeitbar, da sie ausschließlich den ANSI-Zeichensatz verwendet. Die Formatierungen werden über im Text vorkommende Steuersequenzen gesetzt. Jeder Formatbefehl wird immer durch einen Backslash “**”** eingeleitet.



Beispiel:

\rtf1

identifiziert eine RTF-Datei der Version 1.

RTF-Datei-Identifizierung

Alle Formatierungsabschnitte eines Textes sind durch geschweifte Klammern **“{}”** begrenzt. Eine RTF-Datei beginnt immer mit **{** und endet immer mit **}**. Somit läßt sich diese Datei über die ersten sechs Zeichen identifizieren, die immer **{\rtf** plus Versionsziffer lauten müssen.

RTF-Formatierungsbefehle

Von den im RTF definierten Formatierungsbefehlen erkennt KIOSK V1.0 die folgenden:

\rtf*n* .... *identifiziert die Datei als RTF-Datei und gibt die Versionsnummer n des benutzten RTF-Standards an.*

Farbattribute

{**\colorttbl;**
\red*r***\green***g***\blue***b***;**
⋮
} .... *erzeugt eine Farbtabelle für die Textausgabe. Sie besteht aus mindestens einer Farbdefinition. Jede Farbdefinition besteht aus je einer* **\red**-, **\green-** *und* **\blue-** *Anweisung mit abschließendem Semikolon “***;***”. Für r, g und b sind Intensitätswerte im Bereich 0 bis 255 zulässig.*

\cf*n* .... *setzt die Textausgabefarbe auf den Tabellenwert n. Der erste Eintrag einer definierten Farbtabelle hat den Index Eins. Ist für n Null angegeben, wird die*

Absatzattribute *Untergrundfarbe gewählt.*

\li*n* .... *setzt den linken Einzug für alle Zeilen auf n Twips (= n/1440 Zoll).*

\fi*n* .... *setzt den linken Einzug für die erste Zeile eines Absatzes auf n Twips.*

\ri*n* .... *setzt den rechten Einzug für alle Zeilen auf n Twips (= n/1440 Zoll).*

\ql .... *richtet Text linksbündig aus.*

\qr .... *richtet Text rechtsbündig aus.*

\qc .... *zentriert Text (mittig).*

\pard .... *setzt alle Absatzattribute auf Standardwerte:*

- Ausrichtung .... linksbündig.
- Erstzeileneinzug. Null.
- Linker Einzug .... Null.
- Rechter Einzug .. Null.

Zeichenattribute

\b .... *schaltet Fettdruck ein.*

\b0 .... *schaltet Fettdruck aus.*

\i .... *schaltet Kursivdruck ein.*

\i0 .... *schaltet Kursivdruck aus.*

\ul .... *schaltet Unterstreichen ein.*

\ul0 .... *schaltet Unterstreichen aus.*

\strike .... *schaltet Durchstreichen ein.*

\strike0 .... *schaltet Durchstreichen aus.*

\fs*n* .... *setzt die Größe einer Schrift in n halben Punkten.*

\plain .... *setzt alle Zeichenattribute auf Standardwerte:*

- Fettdruck .... aus.
- Kursivdruck .... aus.
- Unterstreichen ... aus.
- Durchstreichen .. aus.
- Schriftgröße .... 10 Pkt.

Ausgabeattribute

\par .... *markiert das Ende eines Absatzes und führt einen Zeilenumbruch aus.*

\line .... *bricht die aktuelle Zeile um, ohne den Absatz zu beenden.*

\'*hh* .... *wandelt die gegebene Hexadezimalzahl hh in ein Zeichen und fügt es in den Text ein. Das Aussehen des Zeichens hängt vom gesetzten Zeichensatz ab.*



\tab .... *fügt einen Tabulator ein.*

Alle weiteren Steuerbefehle werden vom Programm ignoriert. Spalten, Tabellen und Grafiken sind nicht darstellbar. Soll im Text ein Backslash oder eine geschweifte Klammer ausgegeben werden, so muß diesem Zeichen ein weiterer Backslash vorangehen: "****" oder "**\{**".

Nicht im RTF enthaltene Steuerbefehle

KIOSK V1.0 beherrscht zudem noch weitere KIOSK-spezifische Steuerbefehle, die durch ein "@" einzuleiten sind. Dazu gehören:

Farbattribute

@**cp***n* .... *setzt die Untergrundfarbe auf den Tabellenwert n. Der erste Eintrag einer definierten Farbtabelle hat den Index Eins. Ist für n Null angegeben, wird die Untergrundfarbe auf den Standardwert gesetzt.*

@**cb***n* .... *setzt für die Textausgabe die Hintergrundfarbe auf den Tabellenwert n. Der erste Eintrag einer definierten Farbtabelle hat den Index Eins. Ist für n Null angegeben, wird die Untergrundfarbe gewählt.*

Artikeldaten

@**Preis***artnr* .... *ermittelt über eine Treiber-DLL (Dynamic Link Library), die von Ihnen erstellt werden kann, den aktuellen Preis eines Artikels. Dieser wird über seine Kennung artnr identifiziert. Die Konventionen der DLL sind nachfolgend beschrieben.*

@**Bestand***artnr* ...... *ermittelt über die oben erwähnte DLL den aktuellen Bestand eines Artikels, der über artnr identifiziert wird.*

Die nachfolgenden Informationen betreffen die Konventionsbeschreibungen für eine Kommunikation mit anderen Programmen oder Systemen mittels Treiber-DLL. Diese Ausführungen brauchen Sie nur dann zu lesen, wenn Sie der Windows-Programmierung mächtig sind und eine Anpassung an Ihre Anforderungen wünschen. Standardmäßig antwortet das Programm über **@Preis***artnr* und **@Bestand***artnr* mit der Bemerkung "**Bitte erfragen**".

KIOSKDRV.DLL Aufrufkonventionen

Wenn Sie in Ihren Texten eine automatische Aktualisierung von Artikelpreisen und Beständen wünschen, benötigen Sie eine DLL-fähige Programmiersprache wie Borland Pascal für Windows oder Visual C++ und entsprechende Programmierkenntnisse.
Die zu erzeugende DLL muß zwingend den Namen **KIOSKDRV.DLL** erhalten, in der mindestens zwei Prozeduren oder Funktionen mit den Namen
GetItemPrice und **GetItemInvent** als exportierbar deklariert sind. Beiden Routinen werden beim Aufruf zwei Pointer auf Null-terminierte Strings übergeben. Der erste zeigt auf einen String, der die Artikelnummer enthält, der zweite zeigt auf einen 255 Zeichen großen Puffer, der das Ergebnis der Ermittlung aufnimmt. Dieser zurückgegebene String wird in den ausgegebenen Text eingefügt. Nachfolgend sehen Sie ein unter Borland Pascal 7.0 entwickeltes Beispiel einer KIOSKDRV.DLL.

```
{Der Quellcode der installierten KIOSKDRV.DLL        }
{Entwickelt von Percy Wippler                        }
{Programmiersprache Borland Pascal 7.0               }

library KIOSKDRV;

uses Strings;

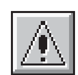
procedure GetItemPrice(ItemNr:PChar;price:PChar);export;
begin
  StrLCopy(price,'Bitte erfragen',14);
end;

procedure GetItemInvent(ItemNr:PChar;invent:PChar);export;
begin
  StrLCopy(invent,'Bitte erfragen',14);
end;

exports GetItemPrice;
exports GetItemInvent;

begin
end.
```

Ein Beispielprogramm für eine KIOSKDRV.DLL



Speichern Sie die compilierte Datei unter dem Namen KIOSKDRV.DLL in Ihrem Windows-Verzeichnis.

5.4.3 Initialisierung

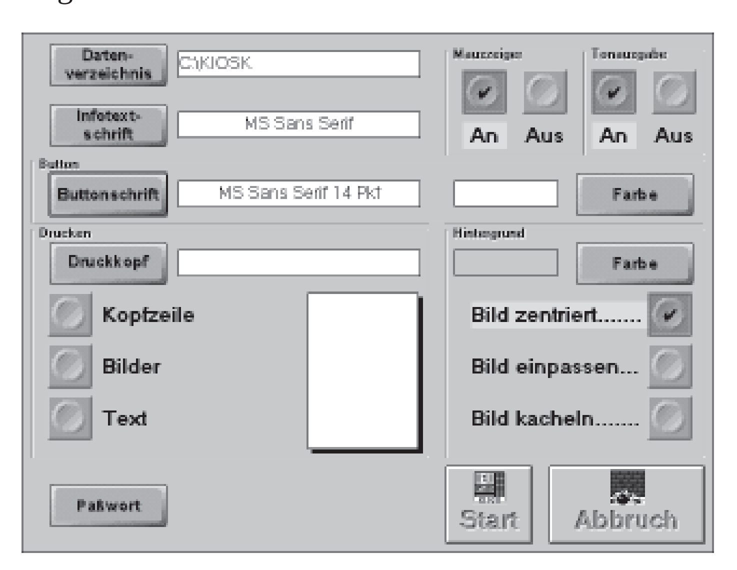
Nachdem Sie KIOSK V1.0 gestartet haben, können Sie wählen, ob das System erst initialisiert oder direkt geladen werden soll. *Abbruch* kehrt zum *Programm-Manager* zurück



Erste Rückfrage nach dem Programmstart

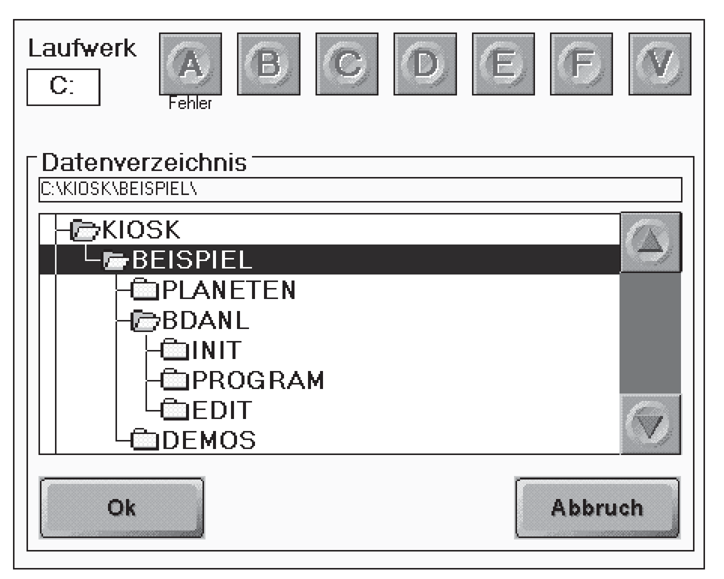
Wenn Sie ***Initialisierung*** wählen, erscheint das nachfolgende Fenster.

Die Initialisierungsmaske



Datenverzeichnis wechseln

Wenn Sie den Button *Datenverzeichnis* betätigen, erscheint ein Dialog, in dem Sie das Laufwerk und das Verzeichnis wechseln können. Den hier gewählten Pfad setzt KIOSK V1.0 als Wurzel aller Präsentationen voraus. Existiert in diesem Verzeichnis keine *INFO.INI*, kann es zu keiner Präsentation kommen. Wählen Sie das Laufwerk über die zur Verfügung stehenden Buttons und das Verzeichnis über die Pfeiltasten. Alle auf dem Datenträger existierenden Verzeichnisse werden aufgelistet. Ist ein Laufwerk nicht lesbar, wird automatisch das nächste angesprochen.

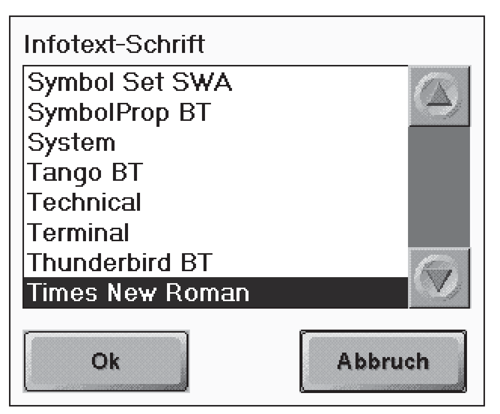


Der Dialog zum Wechsel des Datenverzeichnisses

Betätigen Sie *OK* um das neue Verzeichnis zu übernehmen, oder *Abbruch*, um Ihre Auswahl zu verwerfen.

Infotextschrift wechseln

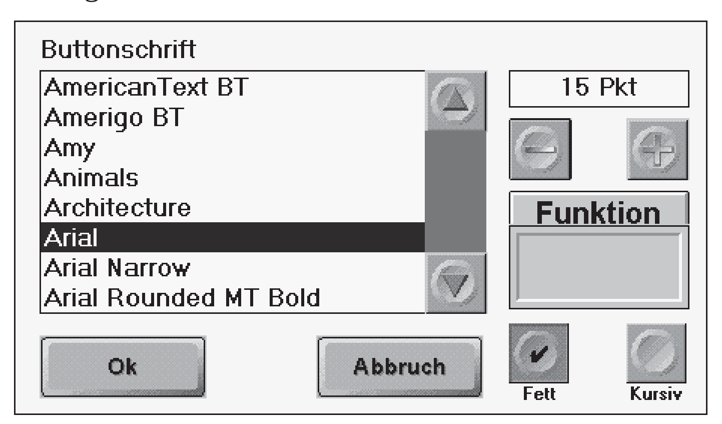
Ist einer Themenpräsentation eine Textdatei hinzugefügt, so werden verschiedene Schriftarten, auch wenn sie im RTF-Text definiert sind, nicht dargestellt. Das Kiosk-System kann nur eine der Schriften darstellen, die auf Ihrem Computer installiert sind. Jedoch können Sie die Schriftart für die Textausgabe Ihren Wünschen anpassen. Betätigen Sie dazu den Button *Infotextschrift*. Im erscheinenden Dialog können Sie über die Pfeiltasten eine Ihrer unter Windows installierten Schriften auswählen. Bestätigen Sie Ihre Wahl mit *OK* oder verwerfen Sie eine Änderung mit *Abbruch*.



Der Dialog zum Schriftwechsel für Textausgaben

Buttonschriftart wechseln

Die Schrift der Schaltflächen läßt sich ebenfalls der Wirkungsweise Ihrer Präsentationen anpassen. Wählen Sie mit dem Button *Buttonschrift* den entsprechenden Dialog.

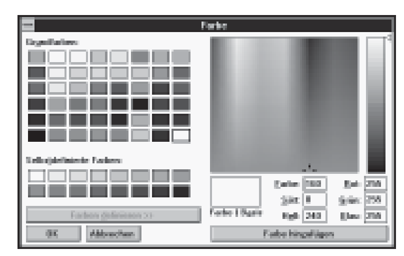


Der Dialog zum Wechseln der Button- und Infozeilenschrift

Sie können über die Pfeiltasten eine Ihrer unter Windows installierten Schriften wählen und deren maximale Schriftgröße über die Plus/Minus-Buttons bestimmen. Zusätzlich können Sie vorgeben, ob die Schaltflächentitel fett oder kursiv ausgegeben werden sollen. Sollte ein Titel breiter als der im Button zur Verfügung stehende Platz sein, wird die Schrift automatisch verkleinert. Ist der Text bei kleinstem Schriftmaß immer noch zu lang, wird er rechts beschnitten.

Buttonfarbe ändern

Nicht nur die Schrift, auch die Farbe der Schaltflächen kann Ihren Anforderungen angepaßt werden. Wenn Sie die Schaltfläche *Farbe* (im Rahmen von *Button*) betätigen, erhalten Sie den Windows-Standarddialog zum Ändern von Farben. Wählen Sie durch einfaches Antippen oder Anklicken eine der vorgegebenen Windows-Grundfarben (links) oder eine aus der gesamten Palette. Diese erscheint, wenn Sie *Farben definieren* betätigen. Die Intensität der Farbe können Sie über die dafür vorgesehene Leiste an der rechten Seite verändern. Bestätigen Sie Ihre Wahl mit *OK* oder verwerfen Sie Ihre Eingaben mit *Abbruch*.



Der Dialog zur Farbauswahl

Hintergrunddarstellung ändern

Neben der Manipulation der Buttons läßt sich der Hintergrund der Darstellungen Ihren Wünschen anpassen. Dazu gehört die Hintergrundfarbe, die genauso ausgewählt wird, wie es unter *Buttonfarbe ändern* beschrieben wurde. Zusätzlich können Bilder, die den Namen ***BACKGR.BMP*** tragen und im *Windows-Bitmap-Format* vorliegen, in den Hintergrund geladen werden. Die Bilddarstellung kann in drei verschiedenen Einstellungen erfolgen.

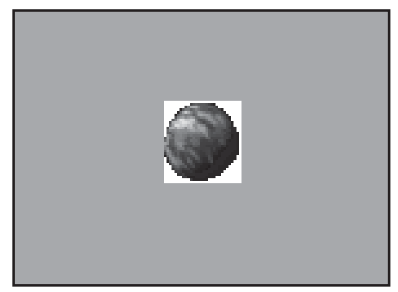


Bild zentriert



Die Darstellungsmöglichkeiten der BACKGR.BMP

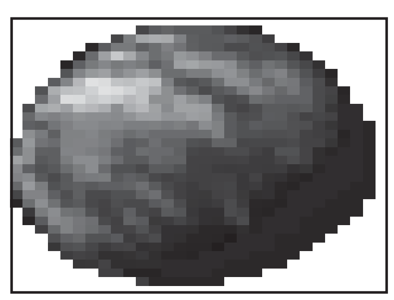


Bild eingepaßt

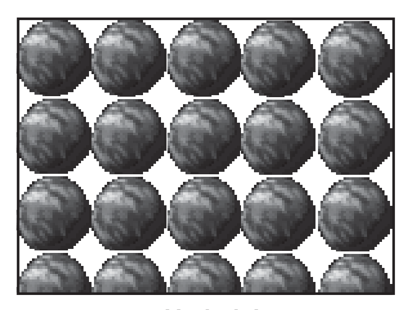


Bild gekachelt

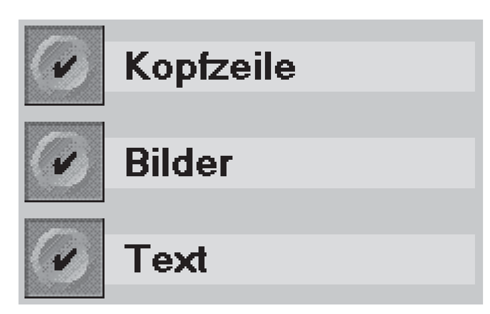
Wie diese drei Modi das Bild darstellen, können Sie den linken Abbildungen entnehmen. Das Hintergrundbild kann bei jedem Verzeichnis neu gesetzt werden, sofern dort eine Datei namens *BACKGR.BMP* existiert. Ist dies nicht der Fall, bleibt das Bild des übergeordneten Verzeichnisses bestehen.
Bei einer Hintergrundbildausgabe wird immer die Farbpalette benutzt, die das System zur Verfügung stellt. Diese reserviert bei einer Videoeinstellung von 256 Farben immer nur 20 Farben für sich. Bilder werden immer nur durch die Systemfarben dargestellt, damit präsentierte Bilder die volle Palette von 256 Farben nutzen können. Es kommt zur Farbreduzierung. Haben Sie allerdings eine Videoeinstellung von mindestens 64K-Farben, kommt es zu keiner Farbreduzierung im Hintergrundbild.

Druckoptionen

Die Daten, die auf dem Bildschirm dargestellt sind, können, sofern es Bilder und/oder Texte sind, auf einem Drucker ausgegeben werden. Voraussetzung dafür ist, daß ein Drucker angeschlossen und installiert ist, dem aktuellen Thema durch den Befehl print=1 eine Druckerlaubnis erteilt wurde und mindestens eine der unter *Drucker* stehenden Optionen ausgewählt ist.



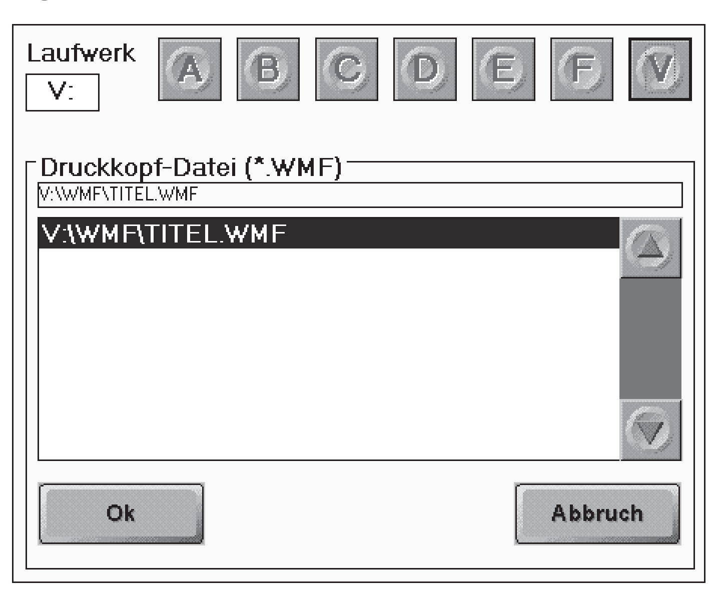
Die ausgewählten Optionen werden auf der Beispielseite angezeigt



Die Auswahl der Druckoptionen

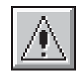
Ist der Button *Text* aktiviert, können Textdateien eines Themas gedruckt werden. Ist *Bilder* aktiviert, können ebenso Grafiken gedruckt werden. Ist *Kopfzeile* ausgewählt, wird in den Kopf eines gedruckten Blattes ein von Ihnen festzulegendes *Windows-Metafile* eingesetzt. Diesen Druckkopf können Sie, nachdem der Button *Druckkopf* betätigt wurde, mit dem folgenden Dialog auswählen.

Wechseln der Druckkopfdatei



Der Dialog zum Wechseln der Druckkopfdatei

Es werden nur die Dateien aufgelistet, die im WMF-Format inkl. Header auf dem von Ihnen gewählten Laufwerk vorhanden sind. Wählen Sie die gewünschte Datei mit den Pfeiltasten und bestätigen Sie Ihre Wahl mit *OK* oder verwerfen Sie die Angaben mit *Abbreuch*

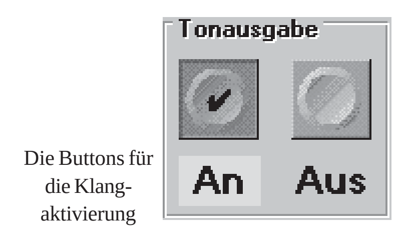


Beachten Sie, daß der Druckkopf immer auf die gesamte Papierbreite gestreckt und in der Höhe angepaßt wird. Daher sollten Sie die WMF-Druckkopfdatei schon in der Druckpapierbreite entwerfen.

Wenn Sie eine Soundkarte installiert haben, aber den-

Tonausgabe ein-/ausschalten

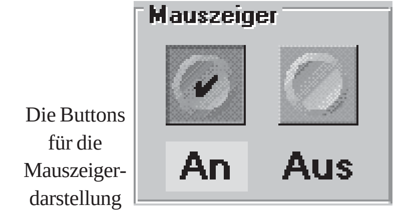
noch eine Tonausgabe vermeiden möchten, können Sie über den Button *Tonausgabe Aus* dafür sorgen, daß zugewiesene Klänge nicht wiedergegeben werden. Möchten Sie die Effekte der Ausgabe von Musik und Sprache nutzen, aktivieren Sie Klänge über *Tonausgabe An.*



Die Buttons für die Klang-aktivierung

Mauszeiger ein-/ausschalten

Wenn Sie KIOSK V1.0 über einen Touchscreen betreiben, ist die Darstellung des Mauszeigers überflüssig. Aus diesem Grund können Sie über die Buttons *An* und *Aus* unter *Mauszeiger* wählen, ob die Maus sichtbar oder unsichtbar sein soll. Während der Initialisierung wird der Cursor als Verbotskreis dargestellt, wenn er im Programm später unsichtbar sein soll.



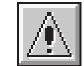
Die Buttons für die Mauszeiger-darstellung

Paßwort ändern

Es besteht die Möglichkeit, das Programm nur nach der Eingabe eines gültigen Paßwortes ablaufen zu lassen. Die Vorgabe dieses Paßwortes erfolgt über die Betätigung des Buttons *Paßwort*. Daraufhin erscheint eine Tastatur, die die Eingabe eines beliebig langen Textes gestattet. Zu Ihrem eigenen Vorteil sollten Sie jedoch nicht mehr als zehn Zeichen verwenden, wobei das Einfügen von Ziffern die Sicherheit erhöht. Nachdem Sie ein neues Paßwort eingegeben haben, werden Sie aufgefordert, dies zur Kontrolle ein zweites Mal zu tun. Wiederholen Sie hier Ihre Eingabe. Wenn beide Eingaben voneinander abweichen, wird das Paßwort gelöscht, und Sie erhalten eine entsprechende Meldung. Ebenso werden Sie informiert, wenn eine Übereinstimmung zutrifft.



Die Tastatur für die Paßworteingabe



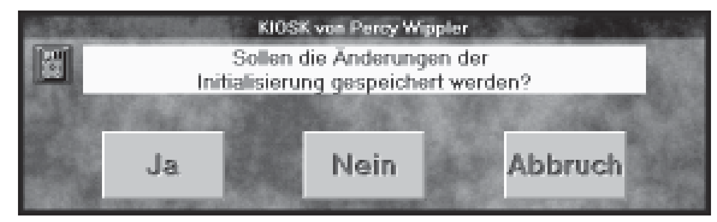
Vergessen Sie nicht, sich das Paßwort irgendwo zu notieren!

Die Buttons *Start* und *Abbruch* beenden die Initialisie-

Initialisierung beenden

rung, wobei nach der Betätigung von *Start* direkt das KIOSK-Programm gestartet wird. Wurden von Ihnen Änderungen der Initialisierung vorgenommen, werden Sie gefragt, ob Sie die neuen Daten speichern möchten.

Initialisierung speichern

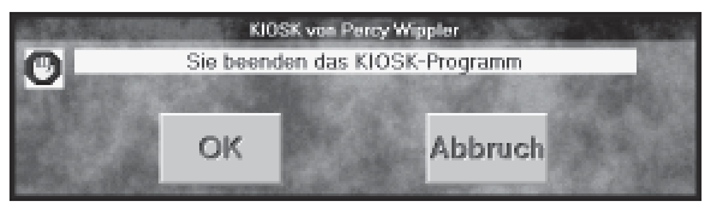


Rückfrage zum Speichern der neuen Vorgaben

Wählen Sie *Ja*, um die neuen Vorgaben zu speichern. Wählen Sie *Nein*, gelten die Daten für den folgenden Start des KIOSK-Programms, werden aber nicht gespeichert. *Abbruch* beendet die Initialisierung nicht.

Wenn Sie das Programm abbrechen, werden Sie zuvor noch gefragt, ob dies wirklich in Ihrer Absicht liegt.

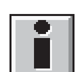
Programm beenden



Rückfrage beim Beenden des Programmes

OK beendet das Programm, *Abbruch* kehrt zum aktuellen Status zurück. Diese Rückfrage erhalten Sie auch, wenn Sie das Kiosk-System verlassen.

Damit ist die Initialisierung abgeschlossen.



Wenn Sie den Abstand der Tasten bei der nebenan gezeigten Tastatur als zu eng empfinden, können Sie durch direkte Manipulation der ***KIOSK.INI*** diesen anpassen. *KIOSK.INI* wurde in Ihrem Windows-Verzeichnis angelegt und läßt sich mit dem Windows-Editor laden und verändern. Geben Sie zur Anpassung die Zeile **KeyDistance=*Pixel*** ein. *Pixel* entspricht einer ganzen Zahl, die die Lücke zwischen den einzelnen Schaltflächen in Pixeln angibt. Von dieser Möglichkeit sollten Sie aber nur Gebrauch machen, wenn Sie einen Touchmonitor und eine höhere Auflösung als 640×480 Pixeln eingestellt haben.

5.5 Benutzung durch den Kunden

Nachdem Sie das Programm aus der Initialisierung oder direkt gestartet haben, besteht die Möglichkeit für den Kunden, die von ihm gewünschten Informationen zu erhalten, sofern Sie diese anbieten. Von nun an ist Ihre Arbeit fast erledigt. Mit Ausnahme des Druckerpapierwechselns bleiben Ihnen noch zwei kleine Aufgaben.

5.5.1 Dem Anbieter vorbehalten
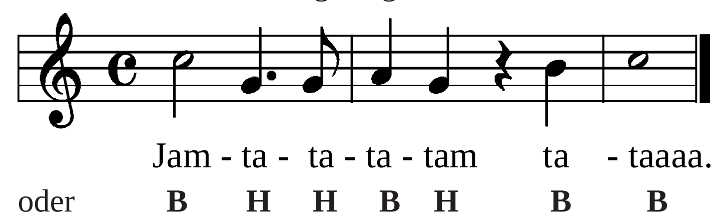
Da das Kiosk-System leicht zu bedienen sein soll, sind die wenigen Funktionen eigentlich selbsterklärend. Allerdings sollten Sie vielleicht am Kiosk-Gehäuse eine Kurzanleitung anbringen, die Sie am Ende dieses Kapitels finden.

Zwei Fragen, die Sie als Anbieter noch beantwortet haben müssen, sind: **Wie beende ich eigentlich das Program?"** und **"Was hat es mit dem *Kommentar* au sich?"**. Die erste Antwort lautet:

Kiosk-System beenden

Wählen Sie die *Übersicht* mittels des entsprechenden Buttons. Es erscheinen die großen grauen Schaltflächen für *Links*- und *Rechtshänder*. Tippen Sie **einmal** auf den Button, dessen Einstellung gerade aktiv ist, tippen Sie **zweimal** auf den Hintergrund, tippen Sie wieder **einmal** auf denselben Button, tippen Sie noch **einmal** auf den Hintergrund und zum Schluß wieder **zweimal** auf den Händerbutton. Ihr Computer wird bei der richtigen Reihenfolge ein Signal von sich geben und Ihnen die vorangehend beschriebene Rückfrage zum *Programm beenden* stellen. Um sich diese Abschaltkombination leichter merken zu können, folgt die Erklärung deren Hintergrundes. Wenn Sie den Film *Roger Rabbit* kennen, entspricht es der Schlagkombination, welcher der Hase nicht widerstehen kann und die ihm sogar fast zum Verhängnis wird. Aber sicherlich ist das nicht der eindeutigste Hinweis. Daher wird hier noch eine bekannte Melodie gezeigt, die dahintersteckt.

Der Abbruch-Code



Hierbei steht B für Button und H für den Hintergrund. Wenn Sie zwischendurch einen anderen Button berühren oder diese Reihenfolge verlassen, müssen Sie von vorn anfangen. Somit ist gewährleistet, daß der Kunde das Programm nicht beenden kann, weil man in aller Regel nicht “willkürlich geordnet” auf dem Bildschirm rumtippt, zumal noch nicht einmal ausschließlich Schaltflächen betroffen sind..

Das Kommentarergebnis

Der Button *Kommentar* bietet dem Kunden die Möglichkeit, einen Einzeiler einzugeben, der stichpunktartig beschreibt, was ihm an Informationen oder Artikeln, bzw. Dienstleistungen Ihres Betriebes fehlt. Diese Eingaben werden mit vorangehendem Datum in einer Datei namens ***VORSCHLG.TXT*** im ANSI-Format gespeichert. Diese Textdatei, die Sie in dem Verzeichnis von *KIOSK.EXE* wiederfinden, können Sie als Anbieter jederzeit z.B. mit dem Windows-Editor lesen und bearbeiten.



Abschließend ist noch zu sagen, daß das Programm in die Übersicht (also Ihr Datenverzeichnis) zurückschaltet, wenn fünf Minuten keine Eingabe erfolgt.

5.5.2 Kurzanleitung

Auf der folgenden Seite finden Sie einen Text, der in dieser oder ähnlicher Form am Kiosk-Gehäuse angebracht werden sollte. Wenn Sie eine eigene Anleitung erstellen möchten, finden Sie die notwendigen Bilder der Buttons im TIF-Format auf Ihrer Installationsdiskette.

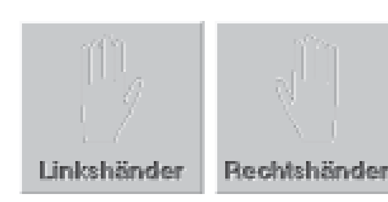
Herzlich willkommen an unserem Kiosk-Informations-System.

Alle Funktionen erfolgen durch einfaches Berühren des Bildschirmes an den entsprechenden Schaltflächen.



Wählen Sie zuerst die **Übersicht**.

Sie haben in der Übersicht die Möglichkeit, alle Elemente so anzuordnen, daß Sie Ihre bevorzugte Hand für die Auswahl besser einsetzen können. Tippen Sie dazu auf eine der beiden dafür vorgesehenen Schaltflächen.



Wählen Sie anschließend bitte eines der dargestellten Themen durch Berührung der entsprechenden Schaltflächen aus.



Wenn die Schaltfläche **Mehr** sichtbar ist, enthält ein Bereich weitere Themen. Berühren Sie **Mehr**, werden diese angezeigt.



Durch Berührung von **Zurück** sehen Sie vorangehende Schaltflächen oder Sie wechseln zu Ihrem zuvor ausgewählten Thema.



Wenn Sie ein bestimmtes Thema suchen, berühren Sie bitte zuerst die Schaltfläche **Suchen**. Anschließend erhalten Sie die Möglichkeit, einen Bereich per Texteingabe zu suchen oder ein Thema durch Betätigung der Pfeiltasten aus den dargestellten Listen zu wählen. Berühren Sie **Gehe Zu**, um das Thema darzustellen, oder **Schließen**, um die Suche abzubrechen.







Ist die Schaltfläche **Drucken** sichtbar, wird Ihnen die aktuelle Information bei Betätigung auf einem Blatt Papier zum Mitnehmen ausgegeben.

Fehlen Ihrer Meinung nach Informationen, Waren oder Dienstleitungen in unserem Geschäft, können Sie uns das über **Kommentar** mitteilen. Geben Sie bitte über die gezeigte Tastatur einen kurzen Text ein und beenden Sie diesen mit **Schließen**.

Ich bestätige, daß ich diese Arbeit selbständig und nur mit den angegebenen Quellen angefertigt habe!

Berlin

Percy Wippler



Literatur-hinweise

1. Rougé, Daniel : Faszination Multimedia
Düsseldorf, SYBEX-Verlag, 1994, 316 S.
ISBN 3-8155-7084-0

2. Buchheit, Marcellus : Windows Programmierbuch
Düsseldorf, SYBEX-Verlag, 1992, 1106 S.
ISBN 3-88745-949-0

3. Neukamp, Gerd-Uwe; Claaßen, André : Das Turbo Pascal für Windows Buch
Düsseldorf, SYBEX-Verlag,1992, 893 S.
ISBN 3-88745-894-X

4. Maslo, Pia; Dillrich, Stefan : Das große Buch zu Visual Basic 3.0 für Windows
Düsseldorf, DATA BECKER, 1993, 900 S.
ISBN 3-89011-636-1

5. Born, Günter : Dateiformate Programmierhandbuch:Algorithmen, Tools und Treiber
München, Addison-Wesley, 1993, 499 S.
ISBN 3-89319-477-0

6. Microsoft : VISUAL BASIC : Professional Features Book
Microsoft Corporation, 1993, 187 S.

7. Borland Pascal 7.0 - Hilfe

8. PC-Bibliothek - Meyers Lexikon
Mannheim, Brockhaus, 1993



Tests

Für einen Systemtest sind folgende Fälle zu prüfen:

Eine INFO.INI mit mehr 30 [Button]-Zuweisungen
Der Button <Mehr> muß sichtbar werden. Über <Mehr> und <Zurück> muß zwischen den Dekaden gewechselt werden können.

Die Buttons erhalten BMP, DIB, RLE, WMF, TIF, PCX und eine TXT-Datei als thumbnail.
Die Dateien BMP, DIB, RLE, WMF müssen im Button eingepaßt sichbar werden. Die anderen Dateien finden keine Berücksichtigung (ohne Systemfehler).

Es sind eine passende und eine zu große Animation zuzuweisen.
Im 1. Fall wird die Animation gespielt, im 2. nicht.

Eine gültige Animation ist ein Mal ohne eine Zuweisung zu **animreplay** und ein Mal mit den Zuweisungen **animreplay=-1, 0, 1 und 10** abzuspielen.
Im 1. Fall muß die Animation genau 1 mal gespielt werden.
Im 2. Fall muß spielt die Animation endlos spielen.
Im 3. Fall darf die Animation nicht abgespielt werden.
Im 4. Fall muß die Animation genau 1 mal gespielt werden.
im 5. Fall muß die Animation genau 10 mal gespielt werden.

sound wird eine ungültige Datei namens error.wav zugewiesen.
Es darf kein Klang gespielt werden (ohne Systemfehler)

Eine gültiger Klang ist ein Mal ohne eine Zuweisung zu **soundreplay** und ein Mal mit den Zuweisungen **soundreplay=-1, 0, 1 und 10** abzuspielen.
Im 1. Fall muß der Klang genau 1 mal gespielt werden.
Im 2. Fall muß spielt der Klang endlos spielen.
Im 3. Fall darf der Klang nicht abgespielt werden.
Im 4. Fall muß der Klang genau 1 mal gespielt werden.
im 5. Fall muß der Klang genau 10 mal gespielt werden (ohne Verzögerung).

Der 5. Fall wird 3 mal mit den Zuweisungen **sounddelay=0, 5, 60** ausgeführt.
Im 1. Fall spielt der Klang 10 mal ohne Verzögerung
Im 2. Fall spielt der Klang 10 mal mit jeweils 5 Sekunden Pause
Im 3. Fall spielt der Klang 10 mal mit jeweils 60 Sekunden Pause
Das erste Abspielen muß unmittelbar ohne Verzögerung ausgeführt werden.

Sounds und Animationen müssen einzeln und parallel mit allen Optionen laufen.

Es ist eine gültige Animation 30 mal abzuspielen. Zusätzlich ist ein gültige Klangdatei endlos mit 5 Sekunden Verzögerung zu spielen und der Option **sound_while_anim=0, 1, 2** mit jeweils **sound_after_anim=0, 1** zuzuordnen. (6 Fälle)
Im 1. Fall wird der Klang alle 5 Sekunden wiederholt (unabhängig von der Animation)
Im 2. Fall startet der Klang synchron mit jedem Animationsbeginn neu.

Im 3. Fall wird der Klang (mit Verzögerung) sooft wiederholt, wie die Animation läuft.
Im 4.-6. Fall spielt der Klang endlos mit 5 Sek. Verzögerung beginnend mit dem Ende der Animation.

Eine gültige Animation und ein gültiges Bild sind zusammen zu initialisieren, die Wiederholungsrate der Animation ist auf 10 zu setzen.
Das Bild darf erst nach Ablauf der 10 Animationen dargestellt werden.

Ein gültiger Klang und ein gültiges Bild sind zu initialisieren. Zusätzlich ist **picture_while_sound=0 und 1** zu setzten.
Im 1. Fall bleibt das Bild bestehen
Im 2. Fall verschwindet das Bild mit dem Ende des Klanges

Der gleiche Test muß mit mehren Wiederholungen und einer beliebigen Verzögerung das gleiche Ergebnis liefern.

Alle Initialisierungsmöglichkeiten sind in einem Fall zu kombinieren. Das Ergebnis muß den Resultaten der Einzeltests entsprechen.

Für gültige Bilddateien sind solche zu wählen die 16, 256 und 16,7 Mil. Farben von 1×1 Pixel bis 800×600 Pixeln Größe reichen. Alle Bilder müssen in Originalgröße bzw. eingepaßt sichtbar werden.

Für Textdateien sind beliebige Formate einzusetzen.
Nur TXT, RTF, BMP und WMF-Dateien werden sichtbar. Zu hohe Dateien müssen in jedem Fall ein erscheinen der Scrollbuttons zur Folge haben. Das Scrollen muß den gesamten Bildbereich zur Darstellung bringen können.

Es ist eine RTF zu integrieren, welche die Kommunikation über KIOSKDRV.DLL nutzt.
Im 1. Fall befindet sich diese DLL im Windowsverzeichnis, im 2. Fall nicht.
Im 1. Fall muß im Text “Bitte erfragen” erscheinen.
Im 2. Fall muß “Fehler” im Text erscheinen.

Es sind mehrere Verzeichnispfade einzurichten, mit INFO.INI-Dateien zu versehen, die über Buttons in diesen Verzeichnissen wechselt. Einige Verzeichnisse erhalten eine BACKGR.BMP-Datei.
Beim Betätigen der Buttons muß in die Verzeichnisse gewechselt werden. Befindet sich in Ihnen eine BACKGR.BMP wird diese im gewählten Modus dargestellt. Befindet sich in im aktuellen Verzeichnis keine, aber in einem Oberverzeichnis diese Datei, wird diese Bild im Hintergrund dargestellt. Beim Weg des Vorgehens muß über <Zurück> nachvollziehbar sein.
Es ist in verschiedene Verzeichnisse mittels <Suchen> zu gehen.
Beim Weg über <Zurück> müssen ebenfalls alle gegangenen Wege abgehbar sein.



Listing

Globale - Deklarationen

```
Declare Sub GetItemPrice Lib "KIOSKDRV.DLL" (ByVal Intemnr$, ByVal price$)
Declare Sub GetItemInvent Lib "KIOSKDRV.DLL" (ByVal Intemnr$, ByVal item$)

Declare Function MedGetDCColors Lib "MEDIA.DLL" (ByVal hDC%) As Integer
Declare Function MedProofAVI Lib "MEDIA.DLL" (ByVal filename$) As Integer
Declare Function MedProofBMP Lib "MEDIA.DLL" (ByVal filename$) As Integer
Declare Function MedProofWAV Lib "MEDIA.DLL" (ByVal filename$) As Integer
Declare Function MedProofWMF Lib "MEDIA.DLL" (ByVal filename$) As Integer
Declare Function MedGetAVISize Lib "MEDIA.DLL" (ByVal filename$, w%, h%) As Integer
Declare Function MedGetBMPSize Lib "MEDIA.DLL" (ByVal filename$, w%, h%, bits%) As Integer
Declare Function MedGetWMFProp Lib "MEDIA.DLL" (ByVal filename$, Height_To_Width As Single) As Integer

Declare Function MedGetDC Lib "MEDIA.DLL" (ByVal hwnd%) As Integer
Declare Sub MedSetBMP Lib "MEDIA.DLL" (ByVal hDC%, ByVal l%, ByVal t%, ByVal w%, ByVal h%, ByVal prop%, ByVal filename$)
Declare Sub MedStretchImg Lib "MEDIA.DLL" (ByVal ShDC%, ByVal sl%, ByVal st%, ByVal sw%, ByVal sh%, ByVal DhDC%, ByVal dl%,
ByVal dt%, ByVal dw%, ByVal dh%, ByVal hwnd%)
Declare Sub MedCopyImg Lib "MEDIA.DLL" (ByVal ShDC%, ByVal sl%, ByVal st%, ByVal sw%, ByVal sh%, ByVal DhDC%, ByVal dx%,
                                            ByVal dy%)
Declare Sub MedMoveImg Lib "MEDIA.DLL" (ByVal ShDC%, ByVal sl%, ByVal st%, ByVal sw%, ByVal sh%, ByVal dx%, ByVal dy%)
Declare Sub MedScrollWnd Lib "MEDIA.DLL" (ByVal hwnd%, ByVal dy%)

Declare Sub MedShadow Lib "MEDIA.DLL" (ByVal hDC%, ByVal l%, ByVal t%, ByVal w%, ByVal h%, ByVal deep%, ByVal intens%)
Declare Sub MedSetWMF Lib "MEDIA.DLL" (ByVal hDC%, ByVal filename$)

Declare Function GetPrivateProfileString Lib "Kernel" (ByVal Appl$, ByVal Key$, ByVal Default$, ByVal Result$, ByVal size%,
                                                        ByVal filename$) As Integer
Declare Function WritePrivateProfileString Lib "Kernel" (ByVal Appl$, ByVal Key$, ByVal text$, ByVal filename$) As Integer

Declare Sub SetWindowPos Lib "User" (ByVal hWnd%, ByVal insert%, ByVal x%, ByVal y%, ByVal cx%, ByVal cy%, ByVal flags%)
Declare Function GetTickCount Lib "USER" () As Long
Declare Function ShowCursor Lib "User" (ByVal OnOff%) As Integer

Global Const datKioskIni = "KIOSK.INI"
Global Const datInfoIni = "INFO.INI"
Global Const datIndexIni = "INDEX.INI"
Global Const datBckBMP = "BACKGR.BMP"
Global Const datPropose = "VORSCHLG.TXT"

Type ThemeRec
   InitName As String
   title As String
   datAnim As String
   animreplay As Integer
   animCounter As Integer
   datPic As String
   datSound As String
   soundreplay As Integer
   soundCounter As Integer
   datText As String
   dir As String
   ItemNr As String
   printable As Integer
End Type

Type PathRec
   path As String
   Button As Integer
End Type

Global ProgramDir As String                         'Verzeichnis, in dem datPropose gespeichert wird(=KIOSK.EXE-Verz.)

Global ExitDelay As Integer
Global ScreenFonts() As String
Global KeyboardMode As Integer

Global StartDir As String

Global fntButton As String
Global fntButtonSize As Integer
Global fntButtonBold As Integer
Global fntButtonItalic As Integer
Global colButton As Long
```

```
Global fntInfoText As String

Global Password As String

Global colHintergrund As Long
    '=TColorRef-Wert

Global modHintergrund As Integer
    '=0 : Bild zentriert
    '=1 : Bild eingepasst
    '=2 : Bild kacheln

Global datDruckkopf As String
Global moddruck As Integer
    'Bit 0-2 auf 1/0 schalten Druckausgabe ein/aus
    'Bit 0 = Kopfzeile
    'Bit 1 = Bilder
    'Bit 2 = Texte

Global modMaus As Integer
    '=0 : keine Maus sichtbar
    '=1 : Maus sichtbar
Global modTon As Integer
    '=0 : keine Tonausgabe
    '=1 : Tonausgabe aktiv

Global modThemePic As Integer
```

Globale - Prozeduren / Funktionen

```
Sub CenterForm (frm As Form)
    frm.Top = (screen.Height - frm.Height) \ 2
    frm.Left = (screen.Width - frm.Width) \ 2
End Sub

Sub delay (msek As Long)
    Dim s As Long

    s = GetTickCount()
    While (GetTickCount() - s < msek) And Not ExitDelay
        DoEvents
    Wend
    ExitDelay = False
End Sub

Function GetINISTring$ (KeyName$, Default$)
    Dim anz%, RetString$

    RetString$ = Space$(50)
    anz% = GetPrivateProfileString("Initialisierung", KeyName$, Default$, RetString$, 50, datKioskIni)
    GetINISTring$ = Left$(RetString$, anz%)
End Function

Sub GetInit ()
    StartDir = UCase$(GetINISTring("StartDir", "C:\KIOSK\"))
    If Right$(StartDir, 1) = "\" Then StartDir = Left$(StartDir, Len(StartDir) - 1)
    fntButton = GetINISTring("ButtonFont", "MS SANS SERIF")
    fntButtonSize = Val(GetINISTring("ButtonFontsize", "14"))
    fntButtonBold = -Val(GetINISTring("ButtonFontBold", "0"))
    fntButtonItalic = -Val(GetINISTring("ButtonFontItalic", "0"))
    colButton = Val(GetINISTring("ButtonColor", "16777215"))
    fntInfoText = GetINISTring("InfotextFont", "MS SANS SERIF")
    colHintergrund = Val(GetINISTring("BackColor", "12632256"))
    modHintergrund = Val(GetINISTring("BackMode", "0"))
    datDruckkopf = GetINISTring("PrintHead", "")
    moddruck = Val(GetINISTring("PrintMode", "0"))
    modMaus = Val(GetINISTring("MouseMode", "1"))
    modTon = Val(GetINISTring("SoundMode", "0"))
End Sub
```

```
Sub Main ()
    Dim i%, code$
    Dim tm As Long

    If Right$(CurDir$, 1) <> "\" Then
        ProgramDir = CurDir$ + "\"
    Else
        ProgramDir = CurDir$
    End If
    Load frmRequest
    GetInit

'Password laden und entschlüsseln
    code$ = GetINISTring("Password", "")
    Password = ""
    If code$ <> "" Then
        For i% = 1 To Len(code$)
            Password = Password + Chr$(Asc(Mid$(code$, i%, 1)) \ (4 - (i% Mod 3)) + 32)
        Next i%
    End If
'Tastatur mit verschlüsselter Ausgabe
    KeyboardMode = 0
'Password abfragen, wenn vorhanden
    Do
        i% = 0
        If Password <> "" Then
            frmKeyboard.txtAnzeige = "Password bitte!"
            frmKeyboard.Tag = ""
            frmKeyboard.Show 1
            If (Password <> frmKeyboard.Tag) Then
                i% = request(2, "Eingabefehler", "Das eingegebene Password ist falsch!", "Schade|Nochmal")
                If i% = 1 Then End
            End If
        End If
    Loop Until i% <> 2
    i% = request(4, "Option", "Welcher Programmteil soll geladen werden?", "Initalisierung|Kiosk|Abbruch")
    Select Case i%
    Case 2
        frmStartup.Visible = True
        frmUser.Visible = True
        frmUser.Refresh
        Unload frmStartup
        Exit Sub
    Case 3: End
    End Select

'Vorbereitungen für frmInit setzen

    frmStartup.Visible = True                           'Programmtitel anzeigen

    ReDim ScreenFonts(screen.FontCount - 1)
    frmInit.Enabled = False
    frmInit.Visible = True
    frmStartup.txtRead.Visible = True
    frmInit.Refresh
    frmStartup.Refresh
    tm = screen.MousePointer
    screen.MousePointer = 11 'Sanduhr
    For i% = 0 To screen.FontCount - 1
        ScreenFonts(i%) = screen.Fonts(i%)
    Next i%
    screen.MousePointer = tm
    frmStartup.txtRead.Visible = False

'Titel zeitverzögert abschalten
    tm = GetTickCount()
    Do
        DoEvents
    Loop Until GetTickCount() > tm + 3000
    Unload frmStartup
End Sub
```

```
Sub mouseVisible (Visible%)
    If Visible Then
        While ShowCursor(1) <= 1: Wend
    Else
        While ShowCursor(0) > -1: Wend
    End If
End Sub

Function request (IconNr%, titel$, txtAnzeige$, Buttons$) As Integer
    Dim ButtonAnz%, TrennPos%, Trenn%, ButtonWidth%, i%, Wi%, abstand%, FormWidth%, ButtonLeft%

'flexible Message-Box
'Parameter:   IconNr%:      0=kein Icon1, 1-3=Vorgabe-Icon1
'             txtAnzeige$:  Request-Text
'             Button$:      Texte der Buttons, getrennt durch |
'Ergebnis:    Nummer des angewählten Buttons

    If IconNr% = 0 Then
        frmRequest.Icon1(0).Visible = False
        frmRequest.txtAnzeige.Left = 8
    Else
        frmRequest.Icon1(0).Picture = frmRequest.Icon1(IconNr% - 1).Picture
    End If

    ButtonAnz% = -1
    TrennPos% = 1
    Do                                          ' Buttons erstellen
        ButtonAnz% = ButtonAnz% + 1
        Trenn% = InStr(TrennPos%, Buttons$, "|")
        If ButtonAnz% Then Load frmRequest.Button(ButtonAnz%)
        If Trenn% = 0 Then Exit Do      ' fertig
        frmRequest.Button(ButtonAnz%).Caption = Mid$(Buttons$, TrennPos%, Trenn% - TrennPos%)
        TrennPos% = Trenn% + 1
    Loop
    frmRequest.Button(ButtonAnz%).Caption = Mid$(Buttons$, TrennPos%)
    ButtonAnz% = ButtonAnz% + 1                 ' Anzahl Buttons
    ButtonWidth% = 0
    For i% = 0 To ButtonAnz% - 1                ' max. Button-Breite ermitteln
        Wi% = frmRequest.TextWidth(frmRequest.Button(i%).Caption)
        If Wi% > ButtonWidth% Then ButtonWidth% = Wi%
    Next i%
    ButtonWidth% = ButtonWidth% + 5

    abstand% = (frmRequest.ScaleWidth - ButtonWidth% * ButtonAnz%) \ (ButtonAnz% + 1)
    FormWidth% = ((ButtonAnz% + 1) * abstand% + ButtonAnz% * ButtonWidth%) * (frmRequest.Width \ frmRequest.ScaleWidth)
    If frmRequest.Width < FormWidth% Then frmRequest.Width = FormWidth%
    frmRequest.txtAnzeige.Width = frmRequest.ScaleWidth - 2 * frmRequest.txtAnzeige.Left
    frmRequest.txtAnzeige.Caption = txtAnzeige$
    frmRequest.Height = (frmRequest.txtAnzeige.Height + 130) * (frmRequest.Height \ frmRequest.ScaleHeight)
    ButtonLeft% = abstand%
    For i% = 0 To ButtonAnz% - 1
        frmRequest.Button(i%).Visible = 1
        frmRequest.Button(i%).Top = frmRequest.ScaleHeight - 65
        frmRequest.Button(i%).Width = ButtonWidth%
        frmRequest.Button(i%).Left = i% * (ButtonWidth% + abstand%) + ButtonLeft%
    Next i%
    frmRequest.txtTitel.Width = frmRequest.ScaleWidth
    frmRequest.txtTitel = titel$

    CenterForm frmRequest
    frmRequest.Show 1
    request = Val(frmRequest.Tag) + 1           ' Ergebnis setzen
End Function

Sub SetINIString (KeyName$, ByVal text$)
    Dim Ret%

    Ret% = WritePrivateProfileString("Initialisierung", KeyName$, text$, datKioskIni)
End Sub
```

Globale - Deklarationen für RTF-Operationen

```
Declare Sub SetBKColor Lib "GDI" (ByVal hDC%, ByVal tcolorref&)

Type RTF_Format
    FontSize As Single
    FontBold As Integer
    FontItalic As Integer
    FontUnderline As Integer
    FontStrikethru As Integer
    ForeColor As Integer
    BackColor As Integer
    Align As Integer          '0=links, 1=rechts, 2=mitte
    FirstLeftMargin As Integer
    LeftMargin As Integer
    RightMargin As Integer
End Type

Global colortable() As Long
Global Tabs() As Integer     'Positionen in Twips, Align
Global TabAlign As Integer   '0=links, 1=rechts, 2=mitte
Global NeuerAbsatz As Integer
Global NeueZeile As Integer

Global HeightOfText As Long
Global StartCopyY As Long
```

Globale - Prozeduren für RTF-Operationen

```
Sub Format_Output (ctrl As Control, txt As String, RTF As RTF_Format)
    Dim i%, x%, y%
    Dim txtRest As String
    Dim txtUmbr As String
    Dim SpcPos As Integer
    Dim CurX As Integer

    On Error GoTo IgnoreOutpErr:

    If ctrl.FontSize <> RTF.FontSize Then
        ctrl.FontSize = RTF.FontSize
        RTF.FontSize = ctrl.FontSize
    End If

    ctrl.FontBold = RTF.FontBold
    ctrl.FontItalic = RTF.FontItalic
    ctrl.FontUnderline = RTF.FontUnderline
    ctrl.FontStrikethru = RTF.FontStrikethru
    If RTF.ForeColor > UBound(colortable) Or RTF.ForeColor < 0 Then RTF.ForeColor = 1
    If RTF.BackColor > UBound(colortable) Or RTF.BackColor < 0 Then RTF.BackColor = 0
    ctrl.ForeColor = colortable(RTF.ForeColor)
    SetBKColor ctrl.hDC, colortable(RTF.BackColor)

    If NeuerAbsatz Then ctrl.CurrentX = RTF.LeftMargin + RTF.FirstLeftMargin
    If NeueZeile Then ctrl.CurrentX = RTF.LeftMargin

    txtRest = txt
    txtUmbr = ""

    'Gesamte Ausgabe ggf. umbrechen
    CurX = ctrl.CurrentX
    While CurX + ctrl.TextWidth(txt) >= ctrl.ScaleWidth - RTF.RightMargin
        txtRest = txt
        'Zeilen kürzen bis sie genau passen
        While CurX + ctrl.TextWidth(txtRest) >= ctrl.ScaleWidth - RTF.RightMargin
            txtRest = Left$(txtRest, Len(txtRest) - 1)
        Wend
        'Zeilen ggf. bis zum vorhergehenden Leerzeichen kürzen
        SpcPos = 0
        i% = 1
        While i% <> 0
            i% = InStr(i% + 1, txtRest, " ")
            If i% <> 0 Then SpcPos = i%
        Wend
        'Wenn ein Leerzeichen vorhanden, Resttext kürzen
```

```
        If SpcPos <> 0 Then txtRest = Left$(txtRest, SpcPos)
        'Zeilenumbruch einfügen
        txtUmbr = txtUmbr + txtRest + Chr$(10)
        txt$ = Mid$(txt, Len(txtRest) + 1, Len(txt))
        CurX = RTF.LeftMargin
    Wend
    txt = txtUmbr + txt

    'linken Einzug bei Zeilenumbrüchen berücksichtigen
    i% = InStr(txt, Chr$(10))
    While i% <> 0
        txtUmbr = Left$(txt, i%)
        Select Case RTF.Align
        Case 1: ctrl.CurrentX = ctrl.ScaleWidth - RTF.RightMargin - ctrl.TextWidth(txtUmbr) - 10
        Case 2: ctrl.CurrentX =  (ctrl.ScaleWidth - RTF.LeftMargin - RTF.RightMargin - ctrl.TextWidth(txtUmbr) + (NeuerAbsatz *
                                 RTF.FirstLeftMargin)) \ 2
        End Select
        ctrl.Print txtUmbr;
        ctrl.CurrentX = RTF.LeftMargin
        txt = Mid$(txt, i% + 1, Len(txt))
        i% = InStr(txt, Chr$(10))
    Wend

    Select Case RTF.Align
    Case 1: ctrl.CurrentX = ctrl.ScaleWidth - RTF.RightMargin - ctrl.TextWidth(txt) - 10
    Case 2: ctrl.CurrentX =   (ctrl.ScaleWidth - RTF.LeftMargin - RTF.RightMargin - ctrl.TextWidth(txt) + (NeuerAbsatz *
                              RTF.FirstLeftMargin)) \ 2
    End Select
    ctrl.Print txt;
    HeightOfText = ctrl.CurrentY * (ctrl.Height / ctrl.ScaleHeight)
    NeuerAbsatz = False
    NeueZeile = False
    Exit Sub
IgnoreOutpErr:
    Resume Next
End Sub

Sub Format_Print (txt As String, RTF As RTF_Format)
    Dim i%, x%, y%
    Dim txtRest As String
    Dim txtUmbr As String
    Dim SpcPos As Integer
    Dim CurX As Integer

    On Error GoTo IgnorePrintErr

    If printer.FontSize <> RTF.FontSize Then
        printer.FontSize = RTF.FontSize
        RTF.FontSize = printer.FontSize
    End If

    printer.FontBold = RTF.FontBold
    printer.FontItalic = RTF.FontItalic
    printer.FontUnderline = RTF.FontUnderline
    printer.FontStrikethru = RTF.FontStrikethru
    printer.FontTransparent = False

    If NeuerAbsatz Then printer.CurrentX = RTF.LeftMargin + RTF.FirstLeftMargin
    If NeueZeile Then printer.CurrentX = RTF.LeftMargin

    txtRest = txt
    txtUmbr = ""

    'Gesamte Ausgabe ggf. umbrechen
    CurX = printer.CurrentX
    While CurX + printer.TextWidth(txt) >= printer.ScaleWidth - RTF.RightMargin
        txtRest = txt
        'Zeilen kürzen bis sie genau passen
        While CurX + printer.TextWidth(txtRest) >= printer.ScaleWidth - RTF.RightMargin
            txtRest = Left$(txtRest, Len(txtRest) - 1)
        Wend
        'Zeilen ggf. bis zum vorhergehenden Leerzeichen kürzen
        SpcPos = 0
        i% = 1
```

```
        While i% <> 0
            i% = InStr(i% + 1, txtRest, " ")
            If i% <> 0 Then SpcPos = i%
        Wend
        'Wenn ein Leerzeichen vorhanden, Resttext kürzen
        If SpcPos <> 0 Then txtRest = Left$(txtRest, SpcPos)
        'Zeilenumbruch einfügen
        txtUmbr = txtUmbr + txtRest + Chr$(10)
        txt$ = Mid$(txt, Len(txtRest) + 1, Len(txt))
        CurX = RTF.LeftMargin
    Wend
    txt = txtUmbr + txt

    'linken Einzug bei Zeilenumbrüchen berücksichtigen
    i% = InStr(txt, Chr$(10))
    While i% <> 0
        txtUmbr = Left$(txt, i%)
        Select Case RTF.Align
        Case 1: printer.CurrentX = printer.ScaleWidth - RTF.RightMargin - printer.TextWidth(txtUmbr) - 10
        Case 2: printer.CurrentX =    (printer.ScaleWidth - RTF.LeftMargin - RTF.RightMargin - printer.TextWidth(txtUmbr) + (NeuerAbsatz *
                                      RTF.FirstLeftMargin)) \ 2
        End Select
        printer.Print txtUmbr;
        printer.CurrentX = RTF.LeftMargin
        txt = Mid$(txt, i% + 1, Len(txt))
        i% = InStr(txt, Chr$(10))
    Wend

    Select Case RTF.Align
    Case 1: printer.CurrentX = printer.ScaleWidth - RTF.RightMargin - printer.TextWidth(txt) - 10
    Case 2: printer.CurrentX = (printer.ScaleWidth - RTF.LeftMargin - RTF.RightMargin - printer.TextWidth(txt) + (NeuerAbsatz *
                               RTF.FirstLeftMargin)) \ 2
    End Select
    printer.Print txt;
    NeuerAbsatz = False
    NeueZeile = False
    Exit Sub
IgnorePrintErr:
    Resume Next

End Sub

Sub GetRTFPart (ctrl As Control, Filenr As Integer, FilePos As Integer, OldFormat As RTF_Format, SendToPrinter As Integer)
    Dim KlammerAuf%, KlammerZu%                 'Vergleicht geöffnete und geschlossene Abschnitte
    Dim BackSlash%   'Zählt die Befehlsposition in einem Abschnitt
    Dim comm As String  'Befehlsbezeichner
    Dim commline As String 'Befehl + Parameter
    Dim param As String  'Parameter
    Dim paramres As Long   'Parameter als Zahl
    Dim outp As String 'Ausgabestring
    Dim Char As String * 1   'Zu bearbeitendes Zeichen
    Dim nChar As String * 1 'Nächstes zeichen
    Dim IsComm As Integer 'Befehl bekannt oder nicht (TRUE|FALSE)
    Dim actFormat As RTF_Format                 'Aktuelle Ausgabe-Formatierung
    Dim StartOfExtraComm As Integer             'Position eines @
    Dim NewTabs As Integer                      'Anzahl der im Abschnitt neu erzeugten Tab-Stops
    Dim i%

    On Error GoTo Ermittlungsfehler

    actFormat = OldFormat  'Format in diesem Bereich übernimmt die vorherige
    BackSlash% = 0
    Get Filenr, FilePos, nChar
    While (nChar <> "}") And Not EOF(Filenr)
        Char = nChar
        FilePos = FilePos + 1
        Get Filenr, FilePos, nChar

        Select Case Char
        Case "{"'Neuer Bereich
            If outp <> "" Then
                If SendToPrinter Then
                    Format_Print outp, actFormat
                Else
                    Format_Output ctrl, outp, actFormat
                End If
```

```
        End If
        outp = ""
        GetRTFPart ctrl, Filenr, FilePos, actFormat, SendToPrinter
        FilePos = FilePos + 1
        Get Filenr, FilePos, nChar
    Case "\" 'Neuer Befehl
        comm = ""
        BackSlash% = BackSlash% + 1
        Do
            FilePos = FilePos + 1
            Char = nChar
            Get #Filenr, FilePos, nChar
            Select Case Char
            Case "\", "{", "}"  'Befehlszeichen werden durch "\" eingeleitet
                nChar = ""
                comm = ""
                outp = outp + Char
                FilePos = FilePos - 1
                Exit Do
            Case Else: If (Char <> Chr$(10)) And Char <> Chr$(13) Then comm = comm + Char        'Befehlermittlung
            End Select
        Loop Until (nChar = "\") Or (nChar = " ") Or (nChar = "{") Or (nChar = "}")

        IsComm = False
'Leerzeichen nach einem Befehl werden übersprungen
        If (nChar = " ") And (Left$(comm, 1) <> "") Then
            FilePos = FilePos + 1
            Get Filenr, FilePos, nChar
        End If
        commline = comm
        comm = UCase$(comm)
'RTFn = Format-Kennung mit Version n
        If Left$(comm, 3) = "RTF" Then IsComm = True

'COLORTBL; = leitet Farbpaletten-Def. ein
        If comm = "COLORTBL;" Then
            IsComm = True
            ReDim Preserve colortable(1)
            comm = ""
            Do
                comm = ""
                Get #Filenr, FilePos + 1, nChar
                Do
                    FilePos = FilePos + 1
                    Char = nChar
                    Get #Filenr, FilePos + 1, nChar
                    If (Char <> Chr$(10)) And Char <> Chr$(13) Then comm = comm + Char
                Loop Until (nChar = "\") Or (nChar = "}")
                comm = UCase$(comm)
'RGB-Werte lesen
                If Left$(comm, 3) = "RED" Then paramres = Val(Mid$(comm, 4, 3))
                If Left$(comm, 5) = "GREEN" Then paramres = paramres + Val(Mid$(comm, 6, 3)) * &H100
                If Left$(comm, 4) = "BLUE" Then paramres = paramres + Val(Mid$(comm, 5, 3)) * &H10000
                If Right$(comm, 1) = ";" Then
                    colortable(UBound(colortable)) = paramres
                    If nChar <> "}" Then ReDim Preserve colortable(UBound(colortable) + 1)
                End If
                FilePos = FilePos + 1
'COLORTBL abgeschlossen
            Loop Until nChar = "}"
        End If

'Pard setzt Absatzattribute auf Standardwerte
        If comm = "PARD" Then
            IsComm = True
            actFormat.FirstLeftMargin = 0
            actFormat.LeftMargin = 0
            actFormat.RightMargin = 0
            actFormat.Align = 0
            ReDim Tabs(1, 0)
            Tabs(0, 0) = -1
            NewTabs = 0
        End If
```

```
'PLAIN setzt Zeichenattribute auf Standartwerte
        If comm = "PLAIN" Then
            IsComm = True
            actFormat.FontSize = 10
            actFormat.FontBold = False
            actFormat.FontItalic = False
            actFormat.FontUnderline = False
            actFormat.FontStrikethru = False
            actFormat.ForeColor = 1
            actFormat.BackColor = 0
        End If

'li setzt den linken Einzug
        If Left$(comm, 2) = "LI" And Mid$(comm, 3, 1) <= "9" Then
            IsComm = True
            param = Mid$(comm, 3, Len(comm))
            actFormat.LeftMargin = Val(param)
        End If

'fi setzt den Einzug der ersten Absatzzeile
        If Left$(comm, 2) = "FI" Then
            IsComm = True
            param = Mid$(comm, 3, Len(comm))
            actFormat.FirstLeftMargin = Val(param)
        End If

'ri setzt den rechten Einzug
        If Left$(comm, 2) = "RI" Then
            IsComm = True
            param = Mid$(comm, 3, Len(comm) - 2)
            actFormat.RightMargin = Val(param)
        End If


'ql setzt Text linksbündig
        If Left$(comm, 2) = "QL" Then
            IsComm = True
            actFormat.Align = 0
        End If

'qr setzt Text linksbündig
        If Left$(comm, 2) = "QR" Then
            IsComm = True
            actFormat.Align = 1
        End If

'qc setzt Text linksbündig
        If Left$(comm, 2) = "QC" Then
            IsComm = True
            actFormat.Align = 2
        End If

'tql setzt linksbündigen Tab-Modus
        If comm = "TQL" Then
            IsComm = True
            TabAlign = 0
        End If

'tqr setzt rechtsbündigen Tab-Modus
        If comm = "TQR" Then
            IsComm = True
            TabAlign = 1
        End If

'tqc setzt mittigen Tab-Modus
        If comm = "TQC" Then
            IsComm = True
            TabAlign = 2
        End If

'tx setzt einen Tab-Stop
        If Left$(comm, 2) = "TX" And Mid$(comm, 3, 1) <= "9" Then
            IsComm = True
            param = Mid$(comm, 3, Len(comm))
            i% = UBound(Tabs, 2) + 1
            ReDim Preserve Tabs(1, i%)
```

```
            NewTabs = NewTabs + 1
            Tabs(0, i% - 1) = Val(param)
            Tabs(1, i% - 1) = TabAlign
            Tabs(0, i%) = -1
            TabAlign = 0
        End If

'tab setzt Tabulator
        If comm = "TAB" Then
            IsComm = True
            outp = outp + Chr$(9)
        End If

'PAR markiert Absatzende (Zeilenumbruch)
        If (comm = "PAR") Then
            IsComm = True
            outp = outp + Chr$(13)
            If SendToPrinter Then
                Format_Print outp, actFormat
            Else
                Format_Output ctrl, outp, actFormat
            End If
            NeuerAbsatz = True
            outp = ""
        End If

'line erzeugt einen Zeilenumbruch
        If comm = "LINE" Then
            IsComm = True
            outp = outp + Chr$(10)
            If SendToPrinter Then
                Format_Print outp, actFormat
            Else
                Format_Output ctrl, outp, actFormat
            End If
            NeueZeile = True
            outp = ""
        End If

'FSn setzt Schriftgröße auf n Pkt
        If (Left$(comm, 2) = "FS") Then
            IsComm = True
            param = Mid$(comm, 3, Len(comm) - 2)
            If NeuerAbsatz Then actFormat.FontSize = Val(param) / 2
        End If

'b schaltet Fettdruck ein
        If comm = "B" Then
            IsComm = True
            actFormat.FontBold = True
        End If

'b0 schaltet Fettdruck aus
        If comm = "B0" Then
            IsComm = True
            actFormat.FontBold = False
        End If

'i schaltet Kursivdruck ein
        If comm = "I" Then
            IsComm = True
            actFormat.FontItalic = True
        End If

'i0 schaltet Kursivdruck aus
        If comm = "I0" Then
            IsComm = True
            actFormat.FontItalic = False
        End If

'ul schaltet Unterstreichen ein
        If comm = "UL" Then
            IsComm = True
            actFormat.FontUnderline = True
        End If
```

```
'ul0 schaltet Unterstreichen aus
        If comm = "UL0" Then
            IsComm = True
            actFormat.FontUnderline = False
        End If

'strike schaltet Durchstreichen ein
        If comm = "STRIKE" Then
            IsComm = True
            actFormat.FontStrikethru = True
        End If

'strike0 schaltet Durchstreichen ein
        If comm = "STRIKE0" Then
            IsComm = True
            actFormat.FontStrikethru = False
        End If

'CFn setzt die Vordergrundfarbe
        If Left$(comm, 2) = "CF" Then
            IsComm = True
            param = Mid$(comm, 3, Len(comm) - 2)
            actFormat.ForeColor = Val(param)
        End If

' ' beschreibt Sonderzeichen durch Hexcode
        If Left$(comm, 1) = "'" Then
            IsComm = True
            param = Mid$(comm, 2, 1)
'obere Tetrade
            paramres = Asc(param) - 48 + (param > "9") * 7
            param = Mid$(comm, 3, 1)
'untere Tetrade
            paramres = paramres * &H10 + (Asc(param) - 48 + (param > "9") * 7)
'Rest anfügen
            outp = outp + Chr$(paramres) + Mid$(commline, 4, Len(commline))
            If nChar = " " Then outp = outp + " "
        End If

'nicht verwendeter Steuerbefehl
        If Not IsComm And (BackSlash% = 1) Then
            KlammerAuf% = 1
            KlammerZu% = 0
            Do
                Get #Filenr, FilePos, nChar
                If nChar = "{" Then KlammerAuf% = KlammerAuf% + 1
                If nChar = "}" Then KlammerZu% = KlammerZu% + 1
                If KlammerAuf% <> KlammerZu% Then FilePos = FilePos + 1
            Loop Until KlammerAuf% = KlammerZu%
        End If
    Case "}"
    Case "@"
        comm = ""
        StartOfExtraComm = FilePos
        Do
            FilePos = FilePos + 1
            Char = nChar
            Get #Filenr, FilePos, nChar
            If (Char <> Chr$(10)) And (Char <> Chr$(13)) Then comm = comm + Char   'Befehlermittlung
        Loop Until (nChar = "\") Or (nChar = " ") Or (nChar = "{") Or (nChar = "}") Or (nChar = "@")
        commline = comm
        comm = UCase$(comm)
        'Schrift-Hintergrundfarbe
        If Left$(comm, 2) = "CB" And Mid$(comm, 3, 1) <= "9" And Mid$(comm, 3, 1) >= "0" Then
            param = Mid$(comm, 3, Len(comm) - 2)
            paramres = Val(param)
            actFormat.BackColor = paramres
            FilePos = StartOfExtraComm + 2 + Len(Trim$(Str$(paramres)))
        End If
        'Papier-Farbe
        If Left$(comm, 2) = "CP" And Mid$(comm, 3, 1) <= "9" And Mid$(comm, 3, 1) >= "0" Then
            param = Mid$(comm, 3, Len(comm) - 2)
            paramres = Val(param)
            colortable(0) = colortable(paramres)
            ctrl.BackColor = colortable(0)
            FilePos = StartOfExtraComm + 2 + Len(Trim$(Str$(paramres)))
```

```
            End If
            If Left$(comm, 5) = "PREIS" Then
                param = Space$(255)
                GetItemPrice Mid$(commline, 6, Len(commline)), param
                param = Trim$(param)
                While Right$(param, 1) = Chr$(0)
                    param = Left$(param, Len(param) - 1)
                Wend
                outp = outp + param
            End If
            If Left$(comm, 7) = "BESTAND" Then
                param = Space$(255)
                GetItemInvent Mid$(commline, 8, Len(commline)), param
                param = Trim$(param)
                While Right$(param, 1) = Chr$(0)
                    param = Left$(param, Len(param) - 1)
                Wend
                outp = outp + param
            End If
        Case Else: If (Char <> Chr$(10)) And (Char <> Chr$(13)) Then outp = outp + Char
        End Select
    Wend
    'Neu erzeugt Tabs löschen
    ReDim Preserve Tabs(1, UBound(Tabs, 2) - NewTabs)
    Tabs(0, UBound(Tabs, 2)) = -1
    If outp <> "" Then
        If SendToPrinter Then
            Format_Print outp, actFormat
        Else
            Format_Output ctrl, outp, actFormat
        End If
    End If

    Exit Sub
Ermittlungsfehler:
    param = " Fehler"
    Resume Next
End Sub
```

frmRequest - Objekte

```
Begin Form frmRequest
   BackColor      =  &H0080C0FF&
   BorderStyle    =  3
   ClientHeight   =  2370
   ClientLeft     =  1650
   ClientTop      =  5490
   ClientWidth    =  8490
   ControlBox     =  0   'False
   Height         =  2715
   Left           =  1620
   MaxButton      =  0   'False
   MinButton      =  0   'False
   Picture        =  FRMREQU.FRX:0000
   ScaleHeight    =  158
   ScaleMode      =  3  'Pixel
   ScaleWidth     =  566
   Top            =  5175
   Width          =  8550
   Begin SSCommand Button
      Caption        =  "Button"
      Font3D         =  4
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  18
      ForeColor      =  &H00404080&
      Height         =  855
      Index          =  0
      Left           =  120
      Outline        =  0   'False
      Top            =  1440
      Width          =  2175
   End
   Begin PictureBox Icon1
      Height         =  510
      Index          =  3
      Left           =  7800
      Picture        =  FRMREQU.FRX:3ADBA
      Top            =  1800
      Visible        =  0   'False
      Width          =  510
   End
   Begin PictureBox Icon1
      Height         =  510
      Index          =  2
      Left           =  7200
      Picture        =  FRMREQU.FRX:3B0BC
      Top            =  1800
      Visible        =  0   'False
      Width          =  510
   End
   Begin PictureBox Icon1
      Height         =  510
      Index          =  1
      Left           =  6600
      Picture        =  FRMREQU.FRX:3B3BE
      Top            =  1800
      Visible        =  0   'False
      Width          =  510
   End
   Begin PictureBox Icon1
      AutoSize       =  -1  'True
      BackColor      =  &H00404080&
      BorderStyle    =  0  'Keine
      Height         =  480
      Index          =  0
      Left           =  120
      Picture        =  FRMREQU.FRX:3B6C0
      Top            =  360
      Width          =  480
   End
   Begin Label txtTitel
      Alignment      =  2  'Mitte
      BackStyle      =  0  'Transparent
      Caption        =  "Titel"
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  9.75
      ForeColor      =  &H00E0FFFF&
      Height         =  255
      Left           =  0
      Top            =  20
      Width          =  7095
   End
   Begin Label txtAnzeige
      Alignment      =  2  'Mitte
      AutoSize       =  -1  'True
      BackColor      =  &H00E0FFFF&
      Caption        =  "Fragetext"
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  12
      ForeColor      =  &H00000080&
      Height         =  300
      Left           =  735
      Top            =  360
      Width          =  2955
      WordWrap       =  -1  'True
   End
End
```

frmRequest - Ereignisse

```
Sub Button_Click (Index As Integer)
   frmRequest.Tag = Str$(Index)
   frmRequest.Hide
End Sub
```

frmKeyboard - Objekte

```
Begin Form frmKeyboard
   BackColor      =  &H00A2BFBE&
   BorderStyle    =  1  'Nicht änderbar, einfach
   ControlBox     =  0  'False
   FontBold       =  0  'False
   FontName       =  "Courier"
   FontSize       =  24
   KeyPreview     =  -1  'True
   MaxButton      =  0  'False
   MinButton      =  0  'False
   Begin PictureBox palette
      Picture     =  FRMKEYB.FRX:0000
      Visible     =  0  'False
   End
   Begin CommandButton butDummy
      Caption     =  " "
      Height      =  195
      Left        =  15000
      Top         =  2400
      Width       =  255
   End
   Begin SSCommand butSpace
      AutoSize    =  2
      BevelWidth  =  0
      Picture     =  FRMKEYB.FRX:0AA2
   End
   Begin SSCommand butSchliessen
      AutoSize    =  2
      BevelWidth  =  0
      Outline     =  0  'False
      Picture     =  FRMKEYB.FRX:3DBC
   End
   Begin SSCommand butBackSpace
      AutoSize    =  2
      BevelWidth  =  0
      Outline     =  0  'False
      Picture     =  FRMKEYB.FRX:50F6
   End
   Begin SSCommand butChar
      AutoSize    =  2
      BevelWidth  =  0
      Font3D      =  3  'Inset w/light shading
      FontBold    =  -1  'True
      FontName    =  "MS Sans Serif"
      FontSize    =  13.5
      ForeColor   =  &H00000000&
      Height      =  645
      Index       =  0
      Outline     =  0  'False
      Picture     =  FRMKEYB.FRX:5B70
      Visible     =  0  'False
      Width       =  630
   End
   Begin Label txtAnzeige
      BackColor   =  &H00404040&
      BorderStyle =  1  'nicht änderbar, einfach
      FontBold    =  0  'False
      FontName    =  "Courier"
      FontSize    =  24
      ForeColor   =  &H0080FF80&
      Height      =  540
      Left        =  0
      TabIndex    =  4
      Top         =  0
      Width       =  12000
   End
End
```

frmKeyboard - Deklarationen

```
Dim KeyboardText As String
Dim Showchar As Integer
Dim touchdown As Integer
Dim wait As Integer
```

frmKeyboard - Ereignisse

```
Sub butBackSpace_Click ()
    butDummy.SetFocus
    If Len(KeyboardText) > 0 Then
        KeyboardText = Left$(KeyboardText, Len(KeyboardText) - 1)
        ShowText KeyboardText
        If (KeyboardMode = 4) Then SearchInList
    End If
End Sub

Sub butBackSpace_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not touchdown Then
        touchdown = True
        On Error GoTo fehler
        While touchdown
            If (Len(KeyboardText) > 0) And (wait = 1) Then
                KeyboardText = Left$(KeyboardText, Len(KeyboardText) - 1)
                ShowText (KeyboardText)
            End If
            If wait = 0 Then
                wait = 1
                delay (600)
            End If
            delay (10)
        Wend
    End If
    Exit Sub
fehler:
    Resume Next
    Exit Sub
End Sub

Sub butBackSpace_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
    If touchdown Then
        touchdown = False
    End If
    exitDelay = True
    wait = 0
End Sub

Sub butChar_Click (Index As Integer)
    butDummy.SetFocus
    KeyboardText = KeyboardText + butChar(Index).Caption
    If (KeyboardMode = 4) Then SearchInList
    ShowText KeyboardText
End Sub

Sub butSchliessen_Click ()
    frmKeyboard.Tag = KeyboardText
    frmKeyboard.Visible = False
    If KeyboardMode = 4 Then frmIndex.Tag = "Cancel"
End Sub

Sub butSpace_Click ()
    butDummy.SetFocus
    KeyboardText = KeyboardText + " "
    If (KeyboardMode = 4) Then SearchInList
    ShowText KeyboardText
End Sub
```

```
Sub Form_KeyPress (KeyAnsi As Integer)
    Dim i%
    KeyAnsi = Asc(UCase$(Chr$(KeyAnsi)))
    Select Case KeyAnsi
    Case 228, 246, 252: KeyAnsi = KeyAnsi - 32
    Case Asc("ß"): butChar_Click 23: butChar_Click 23
    End Select
    Select Case KeyAnsi
    Case 45, 46, 48 To 57, 65 To 90, 196, 214, 220
        For i% = 1 To 41
            If Chr$(KeyAnsi) = butChar(i%).Caption Then butChar_Click i%
        Next i%
    Case 32: butSpace_Click
    Case 8: butBackSpace_Click
    Case 13: butSchliessen_Click
    End Select
End Sub

Sub Form_Load ()
    Dim i%, c$
    Dim abstand As Integer
    Dim Top As Integer
    Dim lft As Integer
    Dim w%, h%

    abstand = Val(GetINISTring("Keydistance", "0"))
    Top = 40
    lft = 5
    w% = butChar(0).Width
    h% = butChar(0).Height

    For i% = 1 To 10
        Load butChar(i%)
        Select Case i%
        Case 1 To 9: c$ = Chr$(i% + 48)
        Case 10: c$ = "0"
        End Select
        butChar(i%).Top = Top
        butChar(i%).Left = lft + (i% - 1) * (w% + abstand)
        butChar(i%).Caption = c$
        butChar(i%).Visible = True
    Next i%
    For i% = 11 To 21
        Load butChar(i%)
        Select Case i%
        Case 11: c$ = "Q"
        Case 12: c$ = "W"
        Case 13: c$ = "E"
        Case 14: c$ = "R"
        Case 15: c$ = "T"
        Case 16: c$ = "Z"
        Case 17: c$ = "U"
        Case 18: c$ = "I"
        Case 19: c$ = "O"
        Case 20: c$ = "P"
        Case 21: c$ = "Ü"
        End Select
        butChar(i%).Top = Top + h% + abstand
        butChar(i%).Left = lft + w% * .6 + (i% - 11) * (w% + abstand)
        butChar(i%).Caption = c$
        butChar(i%).Visible = True
    Next i%
    For i% = 22 To 32
        Load butChar(i%)
        Select Case i%
        Case 22: c$ = "A"
        Case 23: c$ = "S"
        Case 24: c$ = "D"
        Case 25: c$ = "F"
        Case 26: c$ = "G"
        Case 27: c$ = "H"
        Case 28: c$ = "J"
        Case 29: c$ = "K"
        Case 30: c$ = "L"
        Case 31: c$ = "Ö"
        Case 32: c$ = "Ä"
```

```
        End Select
        butChar(i%).Top = Top + 2 * (h% + abstand)
        butChar(i%).Left = lft + w% * .9 + (i% - 22) * (w% + abstand)
        butChar(i%).Caption = c$
        butChar(i%).Visible = True
    Next i%
    For i% = 33 To 41
        Load butChar(i%)
        Select Case i%
        Case 33: c$ = "Y"
        Case 34: c$ = "X"
        Case 35: c$ = "C"
        Case 36: c$ = "V"
        Case 37: c$ = "B"
        Case 38: c$ = "N"
        Case 39: c$ = "M"
        Case 40: c$ = "-"
        Case 41: c$ = "."
        End Select
        butChar(i%).Top = Top + 3 * (h% + abstand)
        butChar(i%).Left = lft + w% * 1.5 + (i% - 33) * (w% + abstand)
        butChar(i%).Caption = c$
        butChar(i%).Visible = True
    Next i%
    butBackSpace.Left = butChar(32).Left
    butBackSpace.Top = butChar(1).Top
    butSpace.Top = Top + 4 * (h% + abstand)
    butSchliessen.Top = butSpace.Top
    butSchliessen.Left = butChar(32).Left + w% - butSchliessen.Width + 2
    butSpace.Left = butSchliessen.Left - butSpace.Width - 20
    h% = frmKeyboard.Width \ frmKeyboard.ScaleWidth
    frmKeyboard.Width = h% * (butChar(32).Left + w% + abstand + 5)
    h% = frmKeyboard.Height \ frmKeyboard.ScaleHeight
    frmKeyboard.Height = h% * (butSpace.Top + w% + abstand + 5)
    frmKeyboard.Top = Screen.Height - frmKeyboard.Height
    frmKeyboard.Left = (Screen.Width - frmKeyboard.Width) \ 2
    txtAnzeige.Width = frmKeyboard.ScaleWidth
    frmKeyboard.FontName = txtAnzeige.FontName
    frmKeyboard.FontSize = txtAnzeige.FontSize
    Showchar = txtAnzeige.Width \ frmKeyboard.TextWidth("A")
    If KeyboardMode = 0 Then txtAnzeige = "Password bitte!"
End Sub

Sub Form_Paint ()
    KeyboardText = frmKeyboard.Tag
    If KeyboardMode <> 0 Then ShowText (KeyboardText)
End Sub

Function GetINISTring$ (KeyName$, Default$)
    Dim anz%, RetString$

    RetString$ = Space$(50)
    anz% = GetPrivateProfileString("Initialisierung", KeyName$, Default$, RetString$, 50, datKioskIni)
    GetINISTring$ = Left$(RetString$, anz%)
End Function

Sub SearchInList ()
    Dim i%, t$

    i% = frmIndex.lstIndex.ListCount
    If i% > 0 Then
        Do
            i% = i% - 1
            t$ = UCase$(Left$(frmIndex.lstIndex.List(i%), Len(KeyboardText)))
            If t$ = "" Then i% = 0
        Loop Until (i% = 0) Or (t$ <= KeyboardText)
        frmIndex.lstIndex.ListIndex = i%
    End If
End Sub

Sub ShowText (ByVal t$)
    If KeyboardMode < 2 Then t$ = String$(Len(t$), Chr$(164))
    t$ = t$ + Chr$(128)
    If Len(t$) >= Showchar Then t$ = Right$(t$, Showchar - 1)
    txtAnzeige.Caption = t$
    txtAnzeige.Refresh
End Sub
```

frmStartup - Objekte

```
Begin Form frmStartup
    BorderStyle     =   0  'Keine
    ControlBox      =   0   'False
    Height          =   3345
    MaxButton       =   0   'False
    MinButton       =   0   'False
    Picture         =   FRMSTART.FRX:0000
    Width           =   4980
    Begin Label txtRead
        AutoSize        =   -1  'True
        BackStyle       =   0  'Transparent
        Caption         =   "Lese Schriften..."
        ForeColor       =   &H00C0FFC0&
        Height          =   195
        Left            =   450
        Top             =   2760
        Width           =   1425
    End
End
```

frmStartup - Ereignisse

```
Sub Form_Click ()
    Unload frmStartUp
End Sub

Sub Form_Load ()
    CenterForm frmStartUp
    SetWindowPos frmStartUp.hWnd, -1, 0, 0, 0, 0, &H40
End Sub
```

frmStartDir - Objekte

```
Begin Form frmStartVerz
   BackColor      =  &H00E0FFFF&
   BorderStyle    =  1
   ControlBox     =  0  'False
   Height         =  5805
   MaxButton      =  0  'False
   MinButton      =  0  'False
   Width          =  5025
   Begin Frame groop
      BackColor      =  &H00E0FFFF&
      Caption        =  "Datenverzeichnis"
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  12
      Height         =  4095
      Left           =  120
      Top            =  1200
      Width          =  4695
      Begin SSCommand butScroll
         BevelWidth     =  0
         Height         =  620
         Index          =  1
         Left           =  3960
         Outline        =  0  'False
         Picture        =  FRMSTDIR.FRX:0000
         Top            =  2520
         Width          =  630
      End
      Begin SSCommand butOk
         AutoSize       =  2
         BevelWidth     =  0
         Height         =  630
         Left           =  120
         Outline        =  0  'False
         Picture        =  FRMSTDIR.FRX:0A7A
         Top            =  3345
         Width          =  1665
      End
      Begin SSCommand butAbbruch
         AutoSize       =  2  'Adjust Button Size To Picture
         BevelWidth     =  0
         Height         =  630
         Left           =  2880
         Outline        =  0  'False
         Picture        =  FRMSTDIR.FRX:2034
         Top            =  3345
         Width          =  1665
      End
      Begin SSCommand butScroll
         BevelWidth     =  0
         Height         =  620
         Index          =  0
         Left           =  3960
         Outline        =  0  'False
         Picture        =  FRMSTDIR.FRX:35EE
         Top            =  705
         Width          =  630
      End
      Begin PictureBox box
         BackColor      =  &H00008000&
         BorderStyle    =  0  'Keine
         Height         =  1335
         Left           =  3960
         Top            =  1305
         Width          =  615
      End
      Begin Outline lstVerz
         FontBold       =  -1  'True
         FontName       =  "MS Sans Serif"
         FontSize       =  12
         Height         =  2415
         Left           =  120
         PictureClosed  =  FRMSTDIR.FRX:4068
         PictureLeaf    =  FRMSTDIR.FRX:42AA
         PictureMinus   =  FRMSTDIR.FRX:44EC
         PictureOpen    =  FRMSTDIR.FRX:472E
         PicturePlus    =  FRMSTDIR.FRX:4970
         Style          =  5
         Top            =  720
         Width          =  4215
      End
      Begin Label txtVerz
         BackStyle      =  0  'Transparent
         BorderStyle    =  1  'nicht änderbar, einfach
         FontBold       =  0  'False
         FontName       =  "MS Sans Serif"
         FontSize       =  8.25
         Height         =  255
         Left           =  120
         Top            =  360
         Width          =  4455
      End
   End
   Begin SSCommand butLaufwerk
      AutoSize       =  2  'Adjust Button Size To Picture
      BevelWidth     =  0
      Caption        =  "A"
      Font3D         =  3  'Inset w/light shading
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  18
      ForeColor      =  &H00007000&
      Height         =  630
      Index          =  0
      Left           =  1440
      Picture        =  FRMSTDIR.FRX:4BB2
      RoundedCorners =  0  'False
      Top            =  120
      Width          =  630
   End
   Begin DriveListBox lstDrive
      Visible        =  0  'False
   End
   Begin Label txtFehler
      Alignment      =  2  'Mitte
      BackStyle      =  0  'Transparent
      Caption        =  "Fehler"
      FontBold       =  0  'False
      FontName       =  "MS Sans Serif"
      FontSize       =  8.25
      Height         =  255
      Index          =  0
      Visible        =  0  'False
      Width          =  630
   End
   Begin Label txtLaufwerk
      Alignment      =  2  'Mitte
      BorderStyle    =  1  'nicht änderbar, einfach
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  12
      Height         =  375
      Left           =  120
      Top            =  480
      Width          =  735
   End
   Begin Label Bezeichnung
      BackStyle      =  0  'Transparent
      Caption        =  "Laufwerk"
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  12
      Height         =  375
      Left           =  120
      Top            =  120
      Width          =  1935
   End
End
```

frmStartDir - Deklarationen

```
Dim Indent%
Dim TouchDown As Integer
Dim wait As Integer

Sub AddVerz (path$)
    Dim verz, i%, j%

    On Error GoTo NoDir
    ChDir (path$)
    lstVerz.AddItem path$
    lstVerz.Indent(lstVerz.ListCount - 1) = Indent%
    Indent% = Indent% + 1
    i% = 0
    verz = Dir("*", 16)
    While verz <> ""
        verz = Dir("*", 16)
        j% = 0
        While j% < i%
            verz = Dir
            j% = j% + 1
        Wend
        If verz <> "" Then
            i% = i% + 1
            If (verz <> ".") And (verz <> "..") Then
                AddVerz (verz)
            End If
        End If
    Wend
    If Len(CurDir$) > 3 Then
        ChDir ".."
        Indent% = Indent% - 1
    End If
    Exit Sub
NoDir:
    Exit Sub
    Resume
End Sub
```

frmStartDir - Ereignisse

```
Sub butAbbruch_Click ()
    frmStartVerz.Visible = False
    frmInit.Enabled = True
End Sub
```

```
Sub butLaufwerk_CLick (index As Integer)
    Dim drive As String
    Dim verz As String
    Dim i%
    Dim ErrorCount%

    On Error GoTo NoNewDrive

SetNewDrive:
    drive = UCase$(Left$(lstDrive.List(index), 2))
    ChDrive drive
    ChDir "\"
    lstDrive.ListIndex = index
    txtFehler(index).Visible = False
    lstVerz.Clear
    lstVerz.Refresh
    txtLaufwerk = "Lese"
    txtLaufwerk.Refresh
    Indent% = 0
    AddVerz ("\")
    If lstVerz.ListCount > 0 Then
        For i% = 0 To lstVerz.ListCount - 1
            lstVerz.Expand(i%) = True
            If lstVerz.HasSubItems(i%) Then lstVerz.PictureType(i%) = 1
            verz = lstVerz.FullPath(i%)
            verz = drive + Mid$(verz, 2, Len(verz))
            If verz = UCase$(StartDir) Then lstVerz.ListIndex = i%
        Next i%
        If drive <> UCase$(Left$(StartDir, 2)) Then lstVerz.ListIndex = 0
    End If
    txtLaufwerk = butLaufWerk(index).Caption + ":"
    lstVerz_Click
    Exit Sub
NoNewDrive:
    ErrorCount% = ErrorCount% + 1
    txtFehler(index).Visible = True
    index = lstDrive.ListIndex
    If ErrorCount% > 1 Then
        index = 0
        While butLaufWerk(index).Caption <> "C"
            index = index + 1
        Wend
    End If
    Resume SetNewDrive
End Sub

Sub butOk_Click ()
    StartDir = Mid$(lstVerz.FullPath(lstVerz.ListIndex), 2, 100)
    StartDir = UCase$(Left$(lstDrive.List(lstDrive.ListIndex), 2)) + StartDir
    frmInit.txtStartVerz.Caption = StartDir
    butAbbruch_Click
End Sub

Sub butScroll_MouseDown (index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not TouchDown Then
        TouchDown = True
        On Error GoTo fehler
        While  TouchDown
            If index = 0 Then
                If lstVerz.ListIndex > 0 Then lstVerz.ListIndex = lstVerz.ListIndex - 1
            Else
                If lstVerz.ListIndex < lstVerz.ListCount - 1 Then lstVerz.ListIndex = lstVerz.ListIndex + 1
            End If
            If wait = 0 Then
                wait = 1
                delay (600)
            End If
            delay (10)
        Wend
        lstVerz_Click
    End If
    Exit Sub
fehler:
    Resume Next
    Exit Sub
```

```
End Sub
Sub butScroll_MouseUp (index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If TouchDown Then TouchDown = False
    ExitDelay = True
    wait = 0
End Sub

Sub Form_Load ()
    Dim i%, s%, t%, l%, w%, h%
    Dim spalten As Integer

    spalten = Val(GetINISTring("DriveColumn", "9"))
    If spalten > lstDrive.ListCount Then spalten = lstDrive.ListCount
    If spalten > 9 Then spalten = 9
    If spalten < 3 Then spalten = 3
    t% = butLaufWerk(0).Top
    l% = butLaufWerk(0).Left
    w% = butLaufWerk(0).Width
    h% = butLaufWerk(0).Height
    l% = l% + (w% + 150)
    s% = 1
    For i% = 1 To lstDrive.ListCount - 1
        Load butLaufWerk(i%)
        Load txtFehler(i%)
        s% = s% + 1
        butLaufWerk(i%).Caption = UCase$(Left$(lstDrive.List(i%), 1))
        If butLaufWerk(i%).Caption = UCase$(Left$(StartDir, 1)) Then butLaufwerk_CLick i%
        butLaufWerk(i%).Left = l%
        l% = l% + (w% + 150)
        butLaufWerk(i%).Top = t%
        butLaufWerk(i%).Visible = True
        txtFehler(i%).Left = butLaufWerk(i%).Left
        txtFehler(i%).Top = butLaufWerk(i%).Top + h% - 30
        If (s% = spalten) And (i% <> lstDrive.ListCount - 1) Then
            s% = 0
            t% = t% + h% + 210
            frmStartVerz.Height = frmStartVerz.Height + h% + 210
            groop.Top = groop.Top + h% + 210
            l% = butLaufWerk(0).Left
        End If
    Next i%
    frmStartVerz.Width = butLaufWerk(0).Left + spalten * (w% + 150)
    groop.Width = frmStartVerz.Width - 240
    txtVerz.Width = groop.Width - 240
    butScroll(0).Left = groop.Width - butScroll(0).Width - 120
    butScroll(1).Left = butScroll(0).Left
    box.Left = butScroll(0).Left
    lstVerz.Width = box.Left + 135
    butAbbruch.Left = box.Left - butAbbruch.Width + box.Width
    CenterForm frmStartVerz
    txtLaufwerk = butLaufWerk(lstDrive.ListIndex).Caption + ":"
End Sub

Sub lstVerz_Click ()
    Dim verz As String

    If lstVerz.ListIndex = -1 Then lstVerz.ListIndex = 0
    verz = Mid$(lstVerz.FullPath(lstVerz.ListIndex), 2, 100)
    verz = UCase$(Left$(lstDrive.List(lstDrive.ListIndex), 2)) + verz + "\"
    txtVerz.Caption = verz
End Sub

Sub lstVerz_PictureClick (ListIndex As Integer)
    lstVerz.ListIndex = ListIndex
    lstVerz_Click
End Sub
```

frmInfoFont - Objekte

```
Begin Form frmInfoFont
  BackColor     = &H00E0FFFF&
  BorderStyle   = 1
  ControlBox    = 0  'False
  Height        = 4215
  MaxButton     = 0  'False
  MinButton     = 0  'False
  ScaleMode     = 3  'Pixel
  Width         = 4770
  Begin SSCommand butScroll
    BevelWidth    = 0
    Height        = 620
    Index         = 1
    Left          = 3960
    Outline       = 0  'False
    Picture       = FRMIFONT.FRX:0000
    Top           = 2290
    Width         = 630
  End
  Begin SSCommand butScroll
    BevelWidth    = 0
    Height        = 620
    Index         = 0
    Left          = 3960
    Outline       = 0  'False
    Picture       = FRMIFONT.FRX:0A7A
    Top           = 480
    Width         = 630
  End
  Begin PictureBox Scrollbar
    BackColor     = &H00008000&
    BorderStyle   = 0  'Keine
    Height        = 1335
    Left          = 3960
    Top           = 1080
    Width         = 615
  End
  Begin SSCommand butAbbruch
    AutoSize      = 2
    BevelWidth    = 0
    Height        = 630
    Left          = 2880
    Outline       = 0  'False
    Picture       = FRMIFONT.FRX:14F4
    Top           = 3120
    Width         = 1665
  End
  Begin SSCommand butOk
    AutoSize      = 2
    BevelWidth    = 0
    Height        = 630
    Left          = 120
    Outline       = 0  'False
    Picture       = FRMIFONT.FRX:2AAE
    Top           = 3120
    Width         = 1665
  End
  Begin ListBox lstFont
    FontBold      = -1  'True
    FontName      = "MS Sans Serif"
    FontSize      = 12
    Height        = 2430
    Left          = 120
    Sorted        = -1  'True
    Top           = 480
    Width         = 4215
  End
  Begin Label Bezeichnung
    BackStyle     = 0  'Transparent
    Caption       = "Infotext-Schrift"
    FontBold      = -1  'True
    FontName      = "MS Sans Serif"
    FontSize      = 12
    Height        = 255
    Left          = 120
    Top           = 120
    Width         = 2415
  End
End
```

frmInfoFont - Deklarationen

```
Dim TouchDown As Integer  'Merker für Scrollen
Dim wait As Integer       '1. Verzögerung beim Scrollen
```

frmInfoFont - Ereignisse

```
Sub butAbbruch_Click ()
    frmInfoFont.Visible = False
    frmInit.Enabled = True
End Sub

Sub butOk_Click ()
    fntInfoText = lstFont.List(lstFont.ListIndex)
    frmInit.txtInfoSchrift.Caption = fntInfoText
    butAbbruch_Click
End Sub

Sub butScroll_MouseDown (index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not TouchDown Then
        TouchDown = True
        On Error GoTo fehler
        While  TouchDown
            If index = 0 Then
                If lstFont.ListIndex > 0 Then lstFont.ListIndex = lstFont.ListIndex - 1
            Else
                If lstFont.ListIndex < lstFont.ListCount - 1 Then lstFont.ListIndex = lstFont.ListIndex + 1
            End If
            If wait = 0 Then
                wait = 1
                delay (600)
            End If
            delay (10)
        Wend
    End If
    Exit Sub
fehler:
    Resume Next
    Exit Sub
End Sub

Sub butScroll_MouseUp (index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If TouchDown Then TouchDown = False
    ExitDelay = True
    wait = 0
End Sub

Sub Form_Load ()
    Dim i%

    If lstFont.ListCount = 0 Then
        For i% = 0 To UBound(ScreenFonts)
            lstFont.AddItem ScreenFonts(i%)
        Next i%
    End If
    lstFont.ListIndex = 0
    If fntInfoText <> "" Then
        For i% = 0 To lstFont.ListCount - 1
            If lstFont.List(i%) = fntInfoText Then
                lstFont.ListIndex = i%
                i% = lstFont.ListCount - 1
            End If
        Next i%
    End If
    CenterForm frmInfoFont
End Sub
```

frmButtonFont - Objekte

```
Begin Form frmButtonFont
   BackColor      =  &H00E0FFFF&
   BorderStyle    =  1
   ControlBox     =  0  'False
   Height         =  4245
   Left           =  1920
   MaxButton      =  0  'False
   MinButton      =  0  'False
   Top            =  2205
   Width          =  6960
   Begin SSRibbon butFett
      BackColor      =  &H00C0C0C0&
      BevelWidth     =  0
      GroupNumber    =  2
      Height         =  600
      Left           =  4920
      Outline        =  0  'False
      PictureDn      =  FRMBFONT.FRX:0000
      PictureDnChange =  0  'Use 'PictureUp' Bitmap Unchanged
      PictureUp      =  FRMBFONT.FRX:0A7A
      RoundedCorners =  0  'False
      Top            =  3000
      Width          =  600
   End
   Begin SSRibbon butKursiv
      BackColor      =  &H00C0C0C0&
      BevelWidth     =  0
      Height         =  600
      Left           =  6120
      Outline        =  0  'False
      PictureDisabled =  FRMBFONT.FRX:14F4
      PictureDn      =  FRMBFONT.FRX:1F6E
      PictureDnChange =  1  'Dither 'PictureUp' Bitmap
      PictureUp      =  FRMBFONT.FRX:29E8
      RoundedCorners =  0  'False
      Top            =  3000
      Width          =  600
   End
   Begin SSPanel demoRahmen
      Alignment      =  8
      BackColor      =  &H00C0C0C0&
      BevelInner     =  1  'Inset
      BevelOuter     =  0  'None
      BevelWidth     =  2
      BorderWidth    =  4
      Font3D         =  0  'None
      Height         =  855
      Left           =  4920
      Outline        =  -1  'True
      RoundedCorners =  0  'False
      Top            =  2040
      Width          =  1815
   End
   Begin SSCommand butMinus
      BevelWidth     =  0
      Height         =  620
      Left           =  4920
      Outline        =  0  'False
      Picture        =  FRMBFONT.FRX:3462
      Top            =  960
      Width          =  630
   End
   Begin SSCommand butPlus
      BevelWidth     =  0
      Height         =  620
      Left           =  6120
      Outline        =  0  'False
      Picture        =  FRMBFONT.FRX:3EDC
      Top            =  960
      Width          =  630
   End
Begin SSCommand butScroll
   BevelWidth     =  0
   Height         =  620
   Index          =  1
   Left           =  4080
   Outline        =  0  'False
   Picture        =  FRMBFONT.FRX:4956
   Top            =  2295
   Width          =  630
End
Begin SSCommand butOk
   AutoSize       =  2
   BevelWidth     =  0
   Height         =  630
   Left           =  240
   Outline        =  0  'False
   Picture        =  FRMBFONT.FRX:53D0
   Top            =  3120
   Width          =  1665
End
Begin SSCommand butAbbruch
   AutoSize       =  2
   BevelWidth     =  0
   Height         =  630
   Left           =  3000
   Outline        =  0  'False
   Picture        =  FRMBFONT.FRX:698A
   Top            =  3120
   Width          =  1665
End
Begin PictureBox Scrollbar
   BackColor      =  &H00008000&
   BorderStyle    =  0  'Keine
   Height         =  1335
   Left           =  4080
   Top            =  1080
   Width          =  615
End
Begin SSCommand butScroll
   BevelWidth     =  0
   Height         =  620
   Index          =  0
   Left           =  4080
   Outline        =  0  'False
   Picture        =  FRMBFONT.FRX:7F44
   Top            =  480
   Width          =  630
End
Begin ListBox lstFont
   FontBold       =  -1  'True
   FontName       =  "MS Sans Serif"
   FontSize       =  12
   Height         =  2430
   Left           =  240
   Sorted         =  -1  'True
   Top            =  480
   Width          =  4215
End
Begin SSPanel txtFontDemo
   Alignment      =  6  'Center - TOP
   BackColor      =  &H00C0C0C0&
   BevelWidth     =  2
   BorderWidth    =  4
   Caption        =  "Funktion"
   Font3D         =  0  'None
   Height         =  495
   Left           =  4920
   Outline        =  -1  'True
   RoundedCorners =  0  'False
   Top            =  1680
   Width          =  1815
End
```

```
  Begin Label txtInfo
    Alignment     =  2  'Mitte
    BackStyle     =  0  'Transparent
    Caption       =  "Kursiv"
    Height        =  195
    Index         =  2
    Left          =  6120
    Top           =  3600
    Width         =  600
  End
  Begin Label txtInfo
    Alignment     =  2  'Mitte
    AutoSize      =  -1  'True
    BackStyle     =  0  'Transparent
    Caption       =  "Fett"
    Height        =  195
    Index         =  1
    Left          =  5040
    Top           =  3600
    Width         =  375
  End
  Begin Label txtInfo
    BackStyle     =  0  'Transparent
    Caption       =  "Buttonschrift"
    FontBold      =  -1  'True
    FontName      =  "MS Sans Serif"
    FontSize      =  12
    Height        =  255
    Index         =  0
    Left          =  240
    Top           =  120
    Width         =  2415
  End
  Begin Label txtFontSize
    Alignment     =  2  'Mitte
    BorderStyle   =  1  'nicht änderbar, einfach
    FontBold      =  -1  'True
    FontName      =  "MS Sans Serif"
    FontSize      =  12
    Height        =  375
    Left          =  4920
    Top           =  480
    Width         =  1815
  End
End
```

frmButtonFont - Deklarationen

```
Dim TouchDown As Integer
Dim wait As Integer
```

frmButtonFont - Ereignisse

```
Sub butAbbruch_Click ()
  frmButtonFont.Visible = False
  frmInit.Enabled = True
End Sub

Sub butFett_Click (Value As Integer)
  txtFontDemo.FontBold = Value
  txtFontDemo_Change
End Sub

Sub butKursiv_Click (Value As Integer)
  txtFontDemo.FontItalic = Value
  txtFontDemo_Change
End Sub

Sub butMinus_Click ()
  If Val(txtFontSize) > 6 Then txtFontSize = Format$(Val(txtFontSize) - 1, "### Pkt")
End Sub
```

```
Sub butOk_Click ()
    fntButton = lstFont.List(lstFont.ListIndex)
    fntButtonSize = Val(txtFontSize)
    fntButtonBold = butFett.Value
    fntButtonItalic = butKursiv.Value
    frmInit.txtButtonSchrift.Caption = fntButton + Format$(fntButtonSize, " ### Pkt")
    butAbbruch_Click
End Sub

Sub butPlus_Click ()
    If Val(txtFontSize) < 40 Then txtFontSize = Format$(Val(txtFontSize) + 1, "### Pkt")
End Sub

Sub butScroll_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not TouchDown Then
        TouchDown = True
        On Error GoTo fehler
        While  TouchDown
            If Index = 0 Then
                If lstFont.ListIndex > 0 Then lstFont.ListIndex = lstFont.ListIndex - 1
            Else
                If lstFont.ListIndex < lstFont.ListCount - 1 Then lstFont.ListIndex = lstFont.ListIndex + 1
            End If
            If wait = 0 Then
                wait = 1
                delay (600)
            End If
            delay (10)
        Wend
    End If
    Exit Sub
fehler:
    Resume Next
    Exit Sub
End Sub

Sub butScroll_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If TouchDown Then
        TouchDown = False
        lstFont_Click
    End If
    exitDelay = True
    wait = 0
End Sub

Sub Form_Load ()
    Dim i%
    Dim ctrl As Control

    If lstFont.ListCount = 0 Then
        For i% = 0 To UBound(ScreenFonts)
            lstFont.AddItem ScreenFonts(i%)
        Next i%
    End If
    lstFont.ListIndex = 0
    If fntButton <> "" Then
        For i% = 0 To lstFont.ListCount - 1
            If lstFont.List(i%) = fntButton Then
                lstFont.ListIndex = i%
                i% = lstFont.ListCount - 1
            End If
        Next i%
    End If
    txtFontDemo.FontBold = fntButtonBold
    txtFontDemo.FontItalic = fntButtonItalic
    txtFontDemo.Width = screen.Width * 3 / 20
    DemoRahmen.Width = txtFontDemo.Width
    frmButtonFont.Width = DemoRahmen.Left + DemoRahmen.Width + 200
    butPlus.Left = DemoRahmen.Left + DemoRahmen.Width - butPlus.Width
    butKursiv.Left = butPlus.Left
    txtInfo(2).Left = butKursiv.Left
    txtFontSize.Width = butPlus.Left + butPlus.Width - txtFontSize.Left
    txtFontSize = Format$(fntButtonSize, "### Pkt")
    butFett.Value = fntButtonBold
    butKursiv.Value = fntButtonItalic
    CenterForm frmButtonFont
End Sub
```

```
Sub lstFont_Click ()
    If Not TouchDown Then
        txtFontDemo.FontName = lstFont.List(lstFont.ListIndex)
        txtFontDemo.FontItalic = butKursiv.Value
        txtFontDemo.FontBold = butFett.Value
        txtFontDemo_Change
    End If
End Sub

Sub txtFontDemo_Change ()
    frmButtonFont.FontName = txtFontDemo.FontName
    frmButtonFont.FontSize = txtFontDemo.FontSize
    frmButtonFont.FontBold = txtFontDemo.FontBold
    frmButtonFont.FontItalic = txtFontDemo.FontItalic
    txtFontDemo.Height = frmButtonFont.TextHeight(txtFontDemo.Caption) + 150
    DemoRahmen.Top = txtFontDemo.Top + txtFontDemo.Height - 10 - 150
    DemoRahmen.Height = screen.Height * 2 / 15 - txtFontDemo.Height + 150
End Sub

Sub txtFontSize_Change ()
    txtFontDemo.FontSize = Val(txtFontSize)
    txtFontDemo_Change
End Sub
```

frmDruckKopf - Objekte

```
Begin Form frmDruckKopf
   BackColor       =   &H00E0FFFF&
   BorderStyle     =   1  'Nicht änderbar, einfach
   ControlBox      =   0   'False
   Height          =   5880
   Left            =   2760
   MaxButton       =   0   'False
   MinButton       =   0   'False
   Top             =   1425
   Width           =   5025
   Begin Frame groop
      BackColor       =   &H00E0FFFF&
      Caption         =   "Druckkopf-Datei (*.WMF)"
      FontBold        =   -1  'True
      FontName        =   "MS Sans Serif"
      FontSize        =   12
      Height          =   4215
      Left            =   120
      Top             =   1200
      Width           =   4695
      Begin SSCommand butScroll
         BevelWidth      =   0
         Height          =   620
         Index           =   1
         Left            =   3960
         Outline         =   0   'False
         Picture         =   FRMPRNHD.FRX:0000
         Top             =   2520
         Width           =   630
      End
      Begin SSCommand butScroll
         BevelWidth      =   0
         Height          =   620
         Index           =   0
         Left            =   3960
         Outline         =   0   'False
         Picture         =   FRMPRNHD.FRX:35EE
         Top             =   705
         Width           =   630
      End
      Begin PictureBox box
         BackColor       =   &H00008000&
         BorderStyle     =   0  'Keine
         Height          =   1335
         Left            =   3960
         Top             =   1305
         Width           =   615
      End
      Begin SSCommand butOk
         AutoSize        =   2
         BevelWidth      =   0
         Height          =   630
         Left            =   120
         Outline         =   0   'False
         Picture         =   FRMPRNHD.FRX:0A7A
         Top             =   3345
         Width           =   1665
      End
      Begin SSCommand butAbbruch
         AutoSize        =   2
         BevelWidth      =   0
         Height          =   630
         Left            =   2880
         Outline         =   0   'False
         Picture         =   FRMPRNHD.FRX:2034
         Top             =   3345
         Width           =   1665
      End
      Begin ListBox lstFile
         FontBold        =   -1  'True
         FontName        =   "MS Sans Serif"
         FontSize        =   12
         Height          =   2430
         Left            =   120
         Top             =   710
         Width           =   4215
      End
      Begin Label txtVerz
         BackStyle       =   0  'Transparent
         BorderStyle     =   1  'nicht änderbar, einfach
         FontBold        =   0  'False
         FontName        =   "MS Sans Serif"
         FontSize        =   8.25
         Height          =   255
         Left            =   120
         Top             =   360
         Width           =   4455
      End
   End
   Begin SSCommand butLaufwerk
      AutoSize        =   2  'Adjust Button Size To Picture
      BevelWidth      =   0
      Caption         =   "A"
      Font3D          =   3  'Inset w/light shading
      FontBold        =   -1  'True
      FontName        =   "MS Sans Serif"
      FontSize        =   18
      ForeColor       =   &H00007000&
      Height          =   630
      Index           =   0
      Left            =   1440
      Picture         =   FRMPRNHD.FRX:4068
      RoundedCorners  =   0   'False
      Top             =   120
      Width           =   630
   End
   Begin DriveListBox lstDrive
      Height          =   315
      Left            =   2280
      Top             =   120
      Visible         =   0   'False
      Width           =   2055
   End
   Begin Label txtFehler
      Alignment       =   2  'Mitte
      BackStyle       =   0  'Transparent
      Caption         =   "Fehler"
      FontBold        =   0  'False
      FontName        =   "MS Sans Serif"
      FontSize        =   8.25
      Height          =   255
      Index           =   0
      Left            =   1440
      Top             =   720
      Visible         =   0   'False
      Width           =   630
   End
   Begin Label txtLaufwerk
      Alignment       =   2  'Mitte
      BorderStyle     =   1  'nicht änderbar, einfach
      FontBold        =   -1  'True
      FontName        =   "MS Sans Serif"
      FontSize        =   12
      Height          =   375
      Left            =   120
      Top             =   480
      Width           =   735
   End
   Begin Label Bezeichnung
      BackStyle       =   0  'Transparent
      Caption         =   "Laufwerk"
      FontBold        =   -1  'True
      FontName        =   "MS Sans Serif"
      FontSize        =   12
      Height          =   375
      Left            =   120
      Top             =   120
      Width           =   1935
   End
End
```

frmDruckKopf - Deklarationen

```
Dim Indent%
Dim TouchDown As Integer
Dim wait As Integer

Sub AddVerz (path$)
    Dim Verz$
    Dim file$
    Dim i%, j%

    On Error GoTo NoDir

    ChDir (path$)
    file$ = Dir$("*.WMF")
    While file <> ""
        If Right$(CurDir$, 1) <> "\" Then
            file$ = CurDir$ + "\" + file$
        Else
            file$ = CurDir$ + file$
        End If
        If MedProofWMF(file$) = 0 Then lstFile.AddItem file$
        file$ = Dir$
    Wend
    Indent% = Indent% + 1
    i% = 0
    Verz$ = Dir$("*", 16)
    While Verz$ <> ""
        Verz$ = Dir$("*", 16)
        j% = 0
        While j% < i%
            Verz$ = Dir$
            j% = j% + 1
        Wend
        If Verz$ <> "" Then
            i% = i% + 1
            If (Verz$ <> ".") And (Verz$ <> "..") Then
                AddVerz (Verz$)
            End If
        End If
    Wend
    If Len(CurDir$) > 3 Then
        ChDir ".."
        Indent% = Indent% - 1
    End If
    Exit Sub
NoDir:
    Exit Sub
    Resume
End Sub
```

frmDruckKopf - Ereignisse

```
Sub butAbbruch_Click ()
    frmDruckKopf.Visible = False
    frmInit.Enabled = True
End Sub

Sub butLaufwerk_CLick (index As Integer)
    Dim drive As String
    Dim Verz As String
    Dim i%
    Dim ErrorCount%

    On Error GoTo NoNewDrive

SetNewDrive:
    drive = UCase$(Left$(lstDrive.List(index), 2))
    ChDrive drive
    ChDir "\"
    lstDrive.ListIndex = index
    txtFehler(index).Visible = False
    lstFile.Clear
    lstFile.Refresh
```

```
    txtLaufwerk = "Lese"
    txtLaufwerk.Refresh
    AddVerz ("\")
    If lstFile.ListCount > 0 Then
        For i% = 0 To lstFile.ListCount - 1
            If UCase$(datDruckKopf) = lstFile.List(i%) Then lstFile.ListIndex = i%
        Next i%
        If drive <> UCase$(Left$(datDruckKopf, 2)) Then lstFile.ListIndex = 0
    End If
    txtLaufwerk = butLaufWerk(index).Caption + ":"
    lstFile_Click
    Exit Sub
NoNewDrive:
    ErrorCount% = ErrorCount% + 1
    txtFehler(index).Visible = True
    index = lstDrive.ListIndex
    If ErrorCount% > 1 Then
        index = 0
        While butLaufWerk(index).Caption <> "C"
            index = index + 1
        Wend
    End If
    Resume SetNewDrive
End Sub

Sub butOk_Click ()
    datDruckKopf = lstFile.List(lstFile.ListIndex)
    frmInit.txtWMFDatei = datDruckKopf
    butAbbruch_Click
End Sub

Sub butScroll_MouseDown (index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not TouchDown Then
        TouchDown = True
        On Error GoTo fehler
        While  TouchDown
            If index = 0 Then
                If lstFile.ListIndex > 0 Then lstFile.ListIndex = lstFile.ListIndex - 1
            Else
                If lstFile.ListIndex < lstFile.ListCount - 1 Then lstFile.ListIndex = lstFile.ListIndex + 1
            End If
            If wait = 0 Then
                wait = 1
                delay (600)
            End If
            delay (10)
        Wend
        lstFile_Click
    End If
    Exit Sub
fehler:
    Resume Next
    Exit Sub
End Sub

Sub butScroll_MouseUp (index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If TouchDown Then TouchDown = False
    ExitDelay = True
    wait = 0
End Sub
```

```
Sub Form_Load ()
    Dim i%, s%, t%, l%, w%, h%
    Dim spalten As Integer

    spalten = Val(GetINISTring("DriveColumn", "9"))
    If spalten > lstDrive.ListCount Then spalten = lstDrive.ListCount
    If spalten > 9 Then spalten = 9
    If spalten < 4 Then spalten = 4
    t% = butLaufWerk(0).Top
    l% = butLaufWerk(0).Left
    w% = butLaufWerk(0).Width
    h% = butLaufWerk(0).Height
    l% = l% + (w% + 150)
    s% = 1
    For i% = 1 To lstDrive.ListCount - 1
        Load butLaufWerk(i%)
        Load txtFehler(i%)
        s% = s% + 1
        butLaufWerk(i%).Caption = UCase$(Left$(lstDrive.List(i%), 1))
        If butLaufWerk(i%).Caption = UCase$(Left$(datDruckKopf, 1)) Then
            butLaufwerk_CLick i%
            txtLaufwerk = butLaufWerk(lstDrive.ListIndex).Caption + ":"
        End If
        butLaufWerk(i%).Left = l%
        l% = l% + (w% + 150)
        butLaufWerk(i%).Top = t%
        butLaufWerk(i%).Visible = True
        txtFehler(i%).Left = butLaufWerk(i%).Left
        txtFehler(i%).Top = butLaufWerk(i%).Top + h% - 30
        If (s% = spalten) And (i% <> lstDrive.ListCount - 1) Then
            s% = 0
            t% = t% + h% + 210
            frmDruckKopf.Height = frmDruckKopf.Height + h% + 210
            groop.Top = groop.Top + h% + 210
            l% = butLaufWerk(0).Left
        End If
    Next i%
    frmDruckKopf.Width = butLaufWerk(0).Left + spalten * (w% + 150)
    groop.Width = frmDruckKopf.Width - 240
    txtVerz.Width = groop.Width - 240
    butScroll(0).Left = groop.Width - butScroll(0).Width - 120
    butScroll(1).Left = butScroll(0).Left
    box.Left = butScroll(0).Left
    lstFile.Width = box.Left + 135
    butAbbruch.Left = box.Left - butAbbruch.Width + box.Width
    CenterForm frmDruckKopf
End Sub

Sub lstFile_Click ()
    txtVerz.Caption = lstFile.List(lstFile.ListIndex)
End Sub

Sub lstFile_PictureClick (ListIndex As Integer)
    lstFile.ListIndex = ListIndex
    lstFile_Click
End Sub
```

frmInit - Objekte

```
Begin Form frmInit
   AutoRedraw     =  -1  'True
   BackColor      =  &H00C0C0C0&
   BorderStyle    =  1
   ControlBox     =  0   'False
   Icon           =  FRMINIT.FRX:0000
   Left           =  285
   MaxButton      =  0   'False
   MinButton      =  0   'False
   ScaleMode      =  3  'Pixel
   Top            =  285
   Begin SSCommand butAbbruch
      AutoSize       =  2  'Adjust Button Size To Pic
      Caption        =  "Abbruch"
      Font3D         =  4  'Inset w/heavy shading
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  18
      ForeColor      =  &H00808080&
      Height         =  1095
      Left           =  7320
      Picture        =  FRMINIT.FRX:0302
      Top            =  6000
      Width          =  2175
   End
   Begin SSCommand butStart
      AutoSize       =  2  'Adjust Button Size To Pic
      Caption        =  "Start"
      Font3D         =  4  'Inset w/heavy shading
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  18
      ForeColor      =  &H00808080&
      Height         =  1095
      Left           =  5880
      Picture        =  FRMINIT.FRX:0906
      Top            =  6000
      Width          =  1215
   End
   Begin SSFrame frameSound
      Caption        =  "Tonausgabe"
      Font3D         =  3  'Inset w/light shading
      ForeColor      =  &H00000000&
      Height         =  1545
      Left           =  7800
      Top            =  150
      Width          =  1695
      Begin SSRibbon butTon
         BevelWidth     =  0
         Height         =  600
         Index          =  1
         Left           =  120
         Outline        =  0   'False
         PictureDisabled =  FRMINIT.FRX:0C08
         PictureDn      =  FRMINIT.FRX:1682
         PictureDnChange =  1
         PictureUp      =  FRMINIT.FRX:20FC
         RoundedCorners =  0   'False
         Top            =  360
         Width          =  600
      End
      Begin SSRibbon butTon
         BevelWidth     =  0
         Height         =  600
         Index          =  0
         Left           =  960
         Outline        =  0   'False
         PictureDisabled =  FRMINIT.FRX:2B76
         PictureDn      =  FRMINIT.FRX:35F0
         PictureDnChange =  1
         PictureUp      =  FRMINIT.FRX:406A
         RoundedCorners =  0   'False
         Top            =  360
         Width          =  600
      End
      Begin Label txtTon
         Alignment      =  2  'Mitte
         BackColor      =  &H0080FF80&
         BackStyle      =  0  'Transparent
         Caption        =  "An"
         FontBold       =  -1  'True
         FontName       =  "MS Sans Serif"
         FontSize       =  13.5
         Height         =  375
         Index          =  1
         Left           =  120
         Top            =  1080
         Width          =  615
      End
      Begin Label txtTon
         Alignment      =  2  'Mitte
         BackColor      =  &H0080FF80&
         BackStyle      =  0  'Transparent
         Caption        =  "Aus"
         FontBold       =  -1  'True
         FontName       =  "MS Sans Serif"
         FontSize       =  13.5
         Height         =  375
         Index          =  0
         Left           =  960
         Top            =  1080
         Width          =  615
      End
   End
   Begin SSFrame frameMaus
      Caption        =  "Mauszeiger"
      Font3D         =  3  'Inset w/light shading
      ForeColor      =  &H00000000&
      Height         =  1545
      Left           =  5880
      Top            =  150
      Width          =  1935
      Begin SSRibbon butMaus
         BevelWidth     =  0
         Height         =  600
         Index          =  0
         Left           =  1080
         Outline        =  0   'False
         PictureDisabled =  FRMINIT.FRX:4AE4
         PictureDn      =  FRMINIT.FRX:555E
         PictureDnChange =  1
         PictureUp      =  FRMINIT.FRX:5FD8
         RoundedCorners =  0   'False
         Top            =  360
         Width          =  600
      End
      Begin SSRibbon butMaus
         BevelWidth     =  0
         Height         =  600
         Index          =  1
         Left           =  240
         Outline        =  0   'False
         PictureDisabled =  FRMINIT.FRX:6A52
         PictureDn      =  FRMINIT.FRX:74CC
         PictureDnChange =  1
         PictureUp      =  FRMINIT.FRX:7F46
         RoundedCorners =  0   'False
         Top            =  360
         Width          =  600
      End
      Begin Label txtMaus
         Alignment      =  2  'Mitte
         BackColor      =  &H0080FF80&
         BackStyle      =  0  'Transparent
         Caption        =  "Aus"
         FontBold       =  -1  'True
         FontName       =  "MS Sans Serif"
         FontSize       =  13.5
         Height         =  375
```

```
      Index          =  0
      Left           =  1080
      Top            =  1080
      Width          =  615
   End
   Begin Label txtMaus
      Alignment      =  2  'Mitte
      BackColor      =  &H0080FF80&
      BackStyle      =  0  'Transparent
      Caption        =  "An"
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  13.5
      Height         =  375
      Index          =  1
      Left           =  240
      TabIndex       =  22
      Top            =  1080
      Width          =  615
   End
End
Begin SSCommand butInfoSchrift
   AutoSize       =  2
   BevelWidth     =  0
   Height         =  630
   Left           =  360
   Outline        =  0  'False
   Picture        =  FRMINIT.FRX:89C0
   Top            =  960
   Width          =  1665
End
Begin SSCommand butStartverz
   AutoSize       =  2
   BevelWidth     =  0
   Height         =  630
   Left           =  360
   Outline        =  0  'False
   Picture        =  FRMINIT.FRX:9F7A
   Top            =  120
   Width          =  1665
End
Begin SSFrame frameDrucken
   Caption        =  "Drucken"
   Font3D         =  3  'Inset w/light shading
   ForeColor      =  &H00000000&
   Height         =  3255
   Left           =  120
   TabIndex       =  6
   Top            =  2640
   Width          =  5535
   Begin PictureBox picDruckDemo
      BackColor      =  &H00E0FFFF&
      FontBold       =  0  'False
      FontName       =  "Small Fonts"
      FontSize       =  6
      ForeColor      =  &H00000000&
      Height         =  615
      Index          =  2
      Left           =  3850
      Top            =  2520
      Visible        =  0  'False
      Width          =  1520
   End
   Begin PictureBox picDruckDemo
      AutoSize       =  -1  'True
      BorderStyle    =  0  'Keine
      Height         =  405
      Index          =  1
      Left           =  4440
      Picture        =  FRMINIT.FRX:B534
      Top            =  1800
      Visible        =  0  'False
      Width          =  315
   End
   Begin PictureBox picDruckDemo
      BorderStyle    =  0  'Keine
      Height         =  615
      Index          =  0
      Left           =  3850
      Top            =  970
      Visible        =  0  'False
      Width          =  1520
   End
   Begin SSRibbon butDruck
      BevelWidth     =  0
      Height         =  600
      Index          =  0
      Left           =  240
      Outline        =  0  'False
      PictureDisabled =  FRMINIT.FRX:BBF6
      PictureDn      =  FRMINIT.FRX:C670
      PictureDnChange =  1
      PictureUp      =  FRMINIT.FRX:D0EA
      RoundedCorners =  0  'False
      Top            =  960
      Width          =  600
   End
   Begin SSRibbon butDruck
      BackColor      =  &H00C0C0C0&
      BevelWidth     =  0
      GroupNumber    =  2
      Height         =  600
      Index          =  1
      Left           =  240
      Outline        =  0  'False
      PictureDisabled =  FRMINIT.FRX:DB64
      PictureDn      =  FRMINIT.FRX:E5DE
      PictureDnChange =  0
      PictureUp      =  FRMINIT.FRX:F058
      RoundedCorners =  0  'False
      Top            =  1680
      Width          =  600
   End
   Begin SSRibbon butDruck
      BackColor      =  &H00C0C0C0&
      BevelWidth     =  0
      GroupNumber    =  3
      Height         =  600
      Index          =  2
      Left           =  240
      Outline        =  0  'False
      PictureDisabled =  FRMINIT.FRX:FAD2
      PictureDn      =  FRMINIT.FRX:1054C
      PictureDnChange =  1
      PictureUp      =  FRMINIT.FRX:10FC6
      RoundedCorners =  0  'False
      Top            =  2400
      Width          =  600
   End
   Begin SSCommand butDruckKopf
      AutoSize       =  2
      BevelWidth     =  0
      Height         =  630
      Left           =  240
      Outline        =  0  'False
      Picture        =  FRMINIT.FRX:11A40
      Top            =  240
      Width          =  1665
   End
   Begin Shape box
      BackStyle      =  1  'Undurchsichtig
      Height         =  2175
      Index          =  0
      Left           =  3840
      Top            =  960
      Width          =  1545
   End
   Begin Label txtDruckoption
      BackColor      =  &H0080FF80&
      BackStyle      =  0  'Transparent
      Caption        =  "  Kopfzeile"
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  13.5
```

```
      Height        =  375
      Index         =  0
      Left          =  720
      Top           =  1080
      Width         =  2895
   End
   Begin Label txtDruckoption
      BackColor     =  &H0080FF80&
      BackStyle     =  0  'Transparent
      Caption       =  "  Bilder"
      FontBold      =  -1  'True
      FontName      =  "MS Sans Serif"
      FontSize      =  13.5
      Height        =  375
      Index         =  1
      Left          =  720
      Top           =  1800
      Width         =  2895
   End
   Begin Label txtDruckoption
      BackColor     =  &H0080FF80&
      BackStyle     =  0  'Transparent
      Caption       =  "  Text"
      FontBold      =  -1  'True
      FontName      =  "MS Sans Serif"
      FontSize      =  13.5
      Height        =  375
      Index         =  2
      Left          =  720
      Top           =  2520
      Width         =  2895
   End
   Begin Label txtWMFDatei
      BorderStyle   =  1
      FontBold      =  0  'False
      FontName      =  "MS Sans Serif"
      FontSize      =  12
      Height        =  375
      Left          =  2040
      Top           =  360
      Width         =  3375
   End
   Begin Shape box
      BackColor     =  &H00000000&
      BackStyle     =  1  'Undurchsichtig
      Height        =  2175
      Index         =  1
      Left          =  3900
      Top           =  1020
      Width         =  1545
   End
End
Begin SSFrame frameHintergrund
   Caption       =  "Hintergrund"
   Font3D        =  3  'Inset w/light shading
   Height        =  3255
   Left          =  5880
   Top           =  2640
   Width         =  3615
   Begin SSRibbon butHintergrund
      BevelWidth    =  0
      Height        =  600
      Index         =  2
      Left          =  2820
      Outline       =  0  'False
      PictureDisabled =  FRMINIT.FRX:12FFA
      PictureDn     =  FRMINIT.FRX:13A74
      PictureDnChange =  1
      PictureUp     =  FRMINIT.FRX:144EE
      RoundedCorners =  0  'False
      Top           =  2400
      Width         =  600
   End
   Begin SSRibbon butHintergrund
      BevelWidth    =  0
      Height        =  600
      Index         =  1
      Left          =  2820
      Outline       =  0  'False
      PictureDisabled =  FRMINIT.FRX:14F68
      PictureDn     =  FRMINIT.FRX:159E2
      PictureDnChange =  0
      PictureUp     =  FRMINIT.FRX:1645C
      RoundedCorners =  0  'False
      Top           =  1680
      Width         =  600
   End
   Begin SSRibbon butHintergrund
      BevelWidth    =  0
      Height        =  600
      Index         =  0
      Left          =  2820
      Outline       =  0  'False
      PictureDisabled =  FRMINIT.FRX:16ED6
      PictureDn     =  FRMINIT.FRX:17950
      PictureDnChange =  1
      PictureUp     =  FRMINIT.FRX:183CA
      RoundedCorners =  0  'False
      Top           =  960
      Width         =  600
   End
   Begin SSCommand butHGFarbe
      AutoSize      =  2
      BevelWidth    =  0
      Height        =  630
      Left          =  1800
      Outline       =  0  'False
      Picture       =  FRMINIT.FRX:18E44
      Top           =  240
      Width         =  1665
   End
   Begin PictureBox picHGFarbe
      BackColor     =  &H00000000&
      Height        =  360
      Left          =  120
      TabIndex      =  2
      Top           =  360
      Width         =  1455
   End
   Begin Label txtHintergrund
      BackColor     =  &H0080FF80&
      BackStyle     =  0  'Transparent
      Caption       =  "Bild kacheln......."
      FontBold      =  -1  'True
      FontName      =  "MS Sans Serif"
      FontSize      =  13.5
      Height        =  375
      Index         =  2
      Left          =  360
      Top           =  2520
      Width         =  2535
   End
   Begin Label txtHintergrund
      BackColor     =  &H0080FF80&
      BackStyle     =  0  'Transparent
      Caption       =  "Bild einpassen..."
      FontBold      =  -1  'True
      FontName      =  "MS Sans Serif"
      FontSize      =  13.5
      Height        =  375
      Index         =  1
      Left          =  360
      Top           =  1800
      Width         =  2535
   End
   Begin Label txtHintergrund
      BackColor     =  &H0080FF80&
      BackStyle     =  0  'Transparent
      Caption       =  "Bild zentriert......."
      FontBold      =  -1  'True
      FontName      =  "MS Sans Serif"
      FontSize      =  13.5
      Height        =  375
      Index         =  0
```

```
      Left          =  360
      Top           =  1080
      Width         =  2535
    End
  End
  Begin CommonDialog stdDialog
  End
  Begin SSFrame frameButton
    Caption         =  "Button"
    Font3D          =  3  'Inset w/light shading
    ForeColor       =  &H00000000&
    Height          =  975
    Left            =  120
    Top             =  1680
    Width           =  9375
    Begin PictureBox picButFarbe
      BackColor     =  &H00000000&
      Height        =  360
      Left          =  5880
      Top           =  360
      Width         =  1455
    End
    Begin SSCommand butButtonSchrift
      AutoSize      =  2
      BevelWidth    =  0
      Height        =  630
      Left          =  240
      Outline       =  0   'False
      Picture       =  FRMINIT.FRX:1A3FE
      Top           =  240
      Width         =  1665
    End
    Begin SSCommand butButFarbe
      AutoSize      =  2
      BevelWidth    =  0
      Height        =  630
      Left          =  7560
      Outline       =  0   'False
      Picture       =  FRMINIT.FRX:1B9B8
      Top           =  240
      Width         =  1665
    End
    Begin Label txtButtonSchrift
      Alignment     =  2  'Mitte
      BorderStyle   =  1
      FontBold      =  0   'False
      FontName      =  "MS Sans Serif"
      FontSize      =  12
      Height        =  375
      Left          =  2040
      Top           =  360
      Width         =  3375
    End
  End
  Begin SSCommand butPassword
    AutoSize        =  2
    BevelWidth      =  0
    Height          =  630
    Left            =  360
    Outline         =  0   'False
    Picture         =  FRMINIT.FRX:1CF72
    Top             =  6240
    Width           =  1665
  End
  Begin Label txtInfoSchrift
    Alignment       =  2  'Mitte
    BorderStyle     =  1
    FontBold        =  0   'False
    FontName        =  "MS Sans Serif"
    FontSize        =  12
    Height          =  375
    Left            =  2160
    Top             =  1080
    Width           =  3375
  End
```

```
    Begin Label txtStartVerz
        BorderStyle   =  1
        FontBold      =  0   'False
        FontName      =  "MS Sans Serif"
        FontSize      =  12
        Height        =  375
        Left          =  2160
        Top           =  240
        Width         =  3375
    End
End
```

frmInit - Deklarationen

```
Dim oldSV$, oldBF$, oldBFS%, oldBFB%, oldBFI%,oldBC&, oldTF$, oldPW$, oldHGF&, oldHGM%, oldDD$, oldDM%, oldMM%, oldTM%
```

frmInit - Prozeduren

```
Sub SaveINI ()
    Dim code$, i%

    'Password verschlüsseln
    code$ = ""
    For i% = 1 To Len(Password)
        code$ = code$ + Chr$((Asc(Mid$(Password, i%, 1)) - 32) * (4 - (i% Mod 3)))
    Next i%
    SetINIString "Password", code$
    SetINIString "StartDir", StartDir
    SetINIString "ButtonFont", fntButton
    SetINIString "ButtonFontsize", Str$(fntButtonSize)
    SetINIString "ButtonFontBold", Str$(-fntButtonBold)
    SetINIString "ButtonFontItalic", Str$(-fntButtonItalic)
    SetINIString "ButtonColor", Str$(colButton)
    SetINIString "InfotextFont", fntInfoText
    SetINIString "BackColor", Str$(colHintergrund)
    SetINIString "BackMode", Str$(modHintergrund)
    SetINIString "PrintHead", datDruckkopf
    SetINIString "PrintMode", Str$(modDruck)
    SetINIString "MouseMode", Str$(modMaus)
    SetINIString "SoundMode", Str$(modTon)
End Sub
```

frmInit - Ereignisse

```
Sub butAbbruch_Click ()
    If Request(1, "KIOSK von Percy Wippler", "Sie beenden das KIOSK-Programm", "Ok|Abbrechen") = 1 Then
        If oldPW$ <> Password Or oldSV$ <> StartDir
                    Or oldBF$ <> fntButton
                    Or oldBFS% <> fntButtonSize
                    Or oldBFI% <> fntButtonItalic
                    Or oldBFB% <> fntButtonBold
                    Or oldBC& <> colButton
                    Or oldTF$ <> fntInfoText
                    Or oldHGF& <> colHintergrund
                    Or oldHGM% <> modHintergrund
                    Or oldDD$ <> datDruckkopf
                    Or oldDM% <> modDruck
                    Or oldMM% <> modMaus
                    Or oldTM% <> modTon Then
            Select Case Request(   3, "KIOSK von Percy Wippler", "Sollen die Änderungen der " + Chr$(10) +
                                   "Initialisierung gespeichert werden?", "Ja|Nein|Abbruch")
            Case 1: SaveINI
            Case 3: Exit Sub
            End Select
        End If
        Unload frmInit
        End
    End If
End Sub

Sub butButFarbe_Click ()
    stdDialog.Flags = 1
    stdDialog.Color = picButFarbe.BackColor
    frmInit.stdDialog.Action = 3
    colButton = stdDialog.Color
    picButFarbe.BackColor = colButton
End Sub

Sub butButtonSchrift_Click ()
    frmButtonFont.Show 1
End Sub
```

```
Sub butDruck_Click (Index As Integer, Value As Integer)
    'Markierung der gewählten Zeile
    txtDruckOption(Index).BackStyle = -Value
    Darstellung im Beispielbild
    Objektgröße und Position wird angepasst
    picDruckDemo(Index).Visible = -Value
    If picDruckDemo(0).Visible Then
        picDruckDemo(1).Top = picDruckDemo(0).Top + picDruckDemo(0).Height + 10
    Else
        picDruckDemo(1).Top = picDruckDemo(0).Top
    End If
    If picDruckDemo(1).Visible Then
        picDruckDemo(2).Top = picDruckDemo(1).Top + picDruckDemo(1).Height
    Else
        If picDruckDemo(0).Visible Then
            picDruckDemo(2).Top = picDruckDemo(0).Top + picDruckDemo(0).Height + 10
        Else
            picDruckDemo(2).Top = picDruckDemo(0).Top
        End If
    End If
    If box(0).Top + box(0).Height - picDruckDemo(2).Top - 20 > 0 Then
        picDruckDemo(2).Height = box(0).Top + box(0).Height - picDruckDemo(2).Top - 20
    Else
        picDruckDemo(2).Height = 0
    End If
    'Bit 0-2 auf 1/0 schalten Druckausgabe ein/aus
    'Bit 0 = Kopfzeile
    'Bit 1 = Bilder
    'Bit 2 = Texte
    If Value Then
        modDruck = modDruck Or (2 ^ Index)
    Else
        modDruck = modDruck And Not (2 ^ Index)
    End If
End Sub

Sub butDruckKopf_Click ()
    frmDruckKopf.Show 1
End Sub

Sub butHGFarbe_Click ()
    stdDialog.Flags = 1
    stdDialog.Color = picHGFarbe.BackColor
    frmInit.stdDialog.Action = 3
    colHintergrund = stdDialog.Color
    picHGFarbe.BackColor = colHintergrund
End Sub

Sub butHintergrund_Click (Index As Integer, Value As Integer)
    Dim i%

    For i% = 0 To 2
        butHintergrund(i%).Enabled = (i% <> Index)
        txtHintergrund(i%).BackStyle = -(i% = Index)
    Next i%
    modHintergrund = Index
    '=0 : Bild zentriert
    '=1 : Bild eingepasst
    '=2 : Bild kacheln
End Sub

Sub butInfoSchrift_Click ()
    frmInfoFont.Show 1
End Sub
```

```
Sub butMaus_Click (Index As Integer, Value As Integer)
    Dim i%

    For i% = 0 To 1
        butMaus(i%).Enabled = (i% <> Index)
        txtMaus(i%).BackStyle = -(i% = Index)
    Next i%
    'MouseVisible (index = 0)
    modMaus = Index
    If modMaus = 1 Then
        screen.MousePointer = 0
    Else
        screen.MousePointer = 12
    End If
End Sub

Sub butPassword_Click ()
    Dim code$, n%

    KeyboardMode = 1
    frmKeyboard.Tag = Password
    frmKeyboard.Show 1
    code$ = frmKeyboard.Tag
    If code$ = "" Then Password = ""
    If code$ <> Password Then
        n% = Request(2, "Paßwort prüfen", "Geben Sie zur Kontrolle das neue " + Chr$(10) + "Passwort bitte noch einmal ein.", " Ok ")
        Password = ""
        frmKeyboard.Tag = Password
        frmKeyboard.Show 1
        Password = frmKeyboard.Tag
        If code$ <> Password Then
            Password = ""
            n% = Request(2, "Paßwortfehler", "Die beiden Eingaben stimmen nicht überein." + Chr$(10) +
                                "Bitte versuchen Sie es erneut.", " Ok ")
        Else
            n% = Request(2, "Paßwort", "Das Paßwort wurde aktualisiert.", "Prima")
        End If
    End If
End Sub

Sub butStart_Click ()
    If oldPW$ <> Password Or oldSV$ <> StartDir
                Or oldBF$ <> fntButton
                Or oldBFS% <> fntButtonSize
                Or oldBFI% <> fntButtonItalic
                Or oldBFB% <> fntButtonBold
                Or oldBC& <> colButton
                Or oldTF$ <> fntInfoText
                Or oldHGF& <> colHintergrund
                Or oldHGM% <> modHintergrund
                Or oldDD$ <> datDruckkopf
                Or oldDM% <> modDruck
                Or oldMM% <> modMaus
                Or oldTM% <> modTon Then
        Select Case Request(    3, "KIOSK von Percy Wippler", "Sollen die Änderungen der " + Chr$(10) +
                                "Initialisierung gespeichert werden?", "Ja|Nein|Abbruch")

        Case 1: SaveINI
        Case 3: Exit Sub
        End Select
    End If

    Unload frmButtonFont
    Unload frmInfoFont
    Unload frmDruckKopf
    Unload frmStartVerz
    Erase Screenfonts
    Unload frmInit
    Load frmUser
    frmUser.Visible = True
End Sub

Sub butStartverz_Click ()
    frmStartVerz.Show 1
End Sub
```

```
Sub butTon_Click (Index As Integer, Value As Integer)
    Dim i%

    For i% = 0 To 1
        butTon(i%).Enabled = (i% <> Index)
        txtTon(i%).BackStyle = -(i% = Index)
    Next i%
    modTon = Index
End Sub

Sub Form_Load ()
    Dim prop As Single
    Dim i%, code$

    txtStartVerz = StartDir
    txtButtonSchrift = fntButton + Str$(fntButtonSize) + " Pkt"
    txtInfoSchrift = fntInfoText
    picButFarbe.BackColor = colButton
    picHGFarbe.BackColor = colHintergrund
    butHintergrund_Click modHintergrund, True
    If MedGetWMFProp(datDruckkopf, prop) = 0 Then
        picDruckDemo(0).Height = picDruckDemo(0).Width * prop
        picDruckDemo(0).Picture = LoadPicture(datDruckkopf)
    Else
        datDruckkopf = ""
        picDruckDemo(0).Height = 0
    End If

    'Druckmodus laden
    For i% = 0 To 2
        If (modDruck And (2 ^ i%)) <> 0 Then
            butDruck(i%).Value = True
            butDruck_Click i%, True
        End If
    Next i%
    txtWMFDatei = datDruckkopf

    'Mausmodus laden
    butMaus_Click modMaus, True

    'Tonmodus laden
    Button_Click modTon, True

    oldPW$ = Password
    oldSV$ = StartDir
    oldBF$ = fntButton
    oldBFS% = fntButtonSize
    oldBFB% = fntButtonBold
    oldBFI% = fntButtonItalic
    oldBC& = colButton
    oldTF$ = fntInfoText
    oldHGF& = colHintergrund
    oldHGM% = modHintergrund
    oldDD$ = datDruckkopf
    oldDM% = modDruck
    oldMM% = modMaus
    oldTM% = modTon
    Width = Width * (640 / ScaleWidth)
    Height = Height * (480 / Scaleheight)
    CenterForm frmInit
End Sub
```

```
Sub picDruckDemo_Paint (Index As Integer)
    If Index = 2 Then
        picDruckDemo(Index).Cls
        picDruckDemo(Index).Print "Dieses Kiosk-System wurde"
        picDruckDemo(Index).Print "von Percy Wippler im"
        picDruckDemo(Index).Print "November 1994 als Diplom-"
        picDruckDemo(Index).Print "arbeit für ein gemein-"
        picDruckDemo(Index).Print "sames Projekt der TFH-"
        picDruckDemo(Index).Print "Berlin und Vobis ent-"
        picDruckDemo(Index).Print "wickelt. Es unterstützt"
        picDruckDemo(Index).Print "Multimedia-Formate, wie"
        picDruckDemo(Index).Print "AVI,WAV,WMF,BMP und"
        picDruckDemo(Index).Print "RTF. Das System wird über"
        picDruckDemo(Index).Print "INI-Dateien konfiguriert,"
        picDruckDemo(Index).Print "die die Ausgaben und Re-"
        picDruckDemo(Index).Print "aktionen steuern. Alle"
        picDruckDemo(Index).Print "darzustellenden Dateien"
        picDruckDemo(Index).Print "liegen in entsprechenden"
        picDruckDemo(Index).Print "Verzeichnissen."
    End If
End Sub

Sub txtWMFDatei_Change ()
    Dim prop As Single

    If MedGetWMFProp(datDruckkopf, prop) = 0 Then
        picDruckDemo(0).Height = picDruckDemo(0).Width * prop
        picDruckDemo(0).Picture = LoadPicture(datDruckkopf)
    Else
        datDruckkopf = ""
        picDruckDemo(0).Height = 0
    End If
    butDruck_Click 0, butDruck(0).Value
End Sub
```

frmIndex - Objekte

```
Begin Form frmIndex
   AutoRedraw     =  -1  'True
   BackColor      =  &H00A2BFBE&
   BorderStyle    =  1
   ControlBox     =  0   'False
   MaxButton      =  0   'False
   MinButton      =  0   'False
   Begin PictureBox palette
      Picture        =  FRMINDEX.FRX:0000
      Visible        =  0   'False
   End
   Begin SSCommand butScrollTheme
      BevelWidth     =  0
      Height         =  615
      Index          =  1
      Outline        =  0   'False
      Picture        =  FRMINDEX.FRX:0AA2
      RoundedCorners =  0   'False
      Width          =  615
   End
   Begin SSCommand butScrollTheme
      BevelWidth     =  0
      Height         =  615
      Index          =  0
      Outline        =  0   'False
      Picture        =  FRMINDEX.FRX:151C
      RoundedCorners =  0   'False
      Width          =  615
   End
   Begin SSCommand butScrollIndex
      BevelWidth     =  0
      Height         =  615
      Index          =  0
      Outline        =  0   'False
      Picture        =  FRMINDEX.FRX:1F96
      RoundedCorners =  0   'False
      Width          =  615
   End
   Begin SSCommand butScrollIndex
      BevelWidth     =  0
      Height         =  615
      Index          =  1
      Outline        =  0   'False
      Picture        =  FRMINDEX.FRX:2A10
      RoundedCorners =  0   'False
      Width          =  615
   End
   Begin SSPanel butGoto
      Alignment      =  4
      BackColor      =  &H0000C0C0&
      BevelInner     =  2  'Raised
      BorderWidth    =  4
      Caption        =  "G   E   H Z E U "
      ForeColor      =  &H00000000&
      RoundedCorners =  0   'False
   End
   Begin PictureBox ScrollBar
      BackColor      =  &H00008000&
      BorderStyle    =  0  'Keine
      Index          =  1
      Width          =  615
   End
   Begin PictureBox ScrollBar
      BackColor      =  &H00008000&
      BorderStyle    =  0  'Keine
      Index          =  0
      Width          =  615
   End
   Begin ListBox lstTheme
      FontBold       =  -1  'True
      FontItalic     =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  12
      Sorted         =  -1  'True
   End
   Begin ListBox lstIndex
      FontBold       =  -1  'True
      FontItalic     =  0   'False
      FontName       =  "MS Sans Serif"
      FontSize       =  12
      Sorted         =  -1  'True
   End
   Begin Label txtFeld
      AutoSize       =  -1  'True
      BackStyle      =  0  'Transparent
      Caption        =  "Themen:"
      FontBold       =  0   'False
      FontName       =  "MS Sans Serif"
      FontSize       =  9.75
      Height         =  240
      Index          =  1
      Left           =  0
   End
   Begin Label txtFeld
      AutoSize       =  -1  'True
      BackStyle      =  0  'Transparent
      Caption        =  "Bereiche:"
      FontBold       =  0   'False
      FontName       =  "MS Sans Serif"
      FontSize       =  9.75
      Height         =  240
      Index          =  0
      Left           =  0
      Top            =  0
   End
End
```

frmIndex - Deklarationen

```
Dim TouchDown As Integer
Dim wait As Integer
Dim IndexThemes() As String
```

frmIndex - Ereignisse

```
Sub butGoto_Click ()
    Dim anz%, RetString$

    RetString$ = Space$(120)
    anz% = GetPrivateProfileString(lstIndex.List(lstIndex.ListIndex), lstTheme.List(lstTheme.ListIndex), "", RetString$, 120, ProgramDir +
                                             datIndexINI)
    frmIndex.Tag = Trim$(Left$(RetString$, anz%))
End Sub

Sub butGoto_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not TouchDown Then
        butGoto.BevelInner = 1
        butGoto.BevelOuter = 1
        butGoto.Move butGoto.Left + 2, butGoto.Top + 2
        TouchDown = True
    End If
End Sub

Sub butGoto_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
    If TouchDown Then
        butGoto.BevelInner = 2
        butGoto.BevelOuter = 2
        butGoto.Move butGoto.Left - 2, butGoto.Top - 2
        TouchDown = False
    End If
End Sub

Sub butScrollIndex_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not TouchDown Then
        TouchDown = True
        On Error GoTo fehlerScrIndex
        While  TouchDown
            If Index = 0 Then
                If lstIndex.ListIndex > 0 Then lstIndex.ListIndex = lstIndex.ListIndex - 1
            Else
                If lstIndex.ListIndex < lstIndex.ListCount - 1 Then lstIndex.ListIndex = lstIndex.ListIndex + 1
            End If
            If wait = 0 Then
                wait = 1
                delay (600)
            End If
            delay (10)
        Wend
    End If
    Exit Sub
fehlerScrIndex:
    Resume Next
    Exit Sub
End Sub

Sub butScrollIndex_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If TouchDown Then TouchDown = False
    ExitDelay = True
    wait = 0
End Sub
```

```
Sub butScrollTheme_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not TouchDown Then
        TouchDown = True
        On Error GoTo fehlerScrTheme
        While  TouchDown
            If Index = 0 Then
                If lstTheme.ListIndex > 0 Then lstTheme.ListIndex = lstTheme.ListIndex - 1
            Else
                If lstTheme.ListIndex < lstTheme.ListCount - 1 Then lstTheme.ListIndex = lstTheme.ListIndex + 1
            End If
            If wait = 0 Then
                wait = 1
                delay (600)
            End If
            delay (10)
        Wend
    End If
    Exit Sub
fehlerScrTheme:
    Resume Next
    Exit Sub
End Sub

Sub butScrollTheme_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If TouchDown Then TouchDown = False
    ExitDelay = True
    wait = 0
End Sub

Sub Form_Load ()
    Dim Filenr As Integer, t$, i%

    frmIndex.Width = frmKeyboard.Width
    frmIndex.Top = 0
    frmIndex.Height = screen.Height - frmKeyboard.Height
    frmIndex.Left = frmKeyboard.Left

    butGoto.Width = butScrollIndex(0).Width
    butGoto.Left = 5

    txtFeld(0).Top = 0
    txtFeld(0).Left = butGoto.Left + butGoto.Width + 5
    lstIndex.Top = txtFeld(0).Height
    lstIndex.Left = txtFeld(0).Left
    lstIndex.Height = (frmIndex.ScaleHeight - 20) / 2
    lstIndex.Width = frmIndex.ScaleWidth - 80
    txtFeld(1).Top = lstIndex.Top + lstIndex.Height
    txtFeld(1).Left = txtFeld(0).Left
    lstTheme.Height = lstIndex.Height
    lstTheme.Width = lstIndex.Width
    lstTheme.Left = lstIndex.Left
    lstTheme.Top = txtFeld(1).Top + txtFeld(1).Height
    ScrollBar(0).Left = frmIndex.ScaleWidth - 10 - butScrollIndex(0).Width
    ScrollBar(1).Left = ScrollBar(0).Left
    ScrollBar(0).Top = lstIndex.Top
    ScrollBar(0).Height = lstIndex.Height
    ScrollBar(1).Top = lstTheme.Top
    ScrollBar(1).Height = lstTheme.Height
    butScrollIndex(0).Top = ScrollBar(0).Top
    butScrollIndex(0).Left = ScrollBar(0).Left
    butScrollIndex(1).Top = ScrollBar(0).Top + ScrollBar(0).Height - butScrollIndex(1).Height
    butScrollIndex(1).Left = ScrollBar(0).Left

    butScrollTheme(0).Top = ScrollBar(1).Top
    butScrollTheme(0).Left = ScrollBar(1).Left
    butScrollTheme(1).Top = ScrollBar(1).Top + ScrollBar(0).Height - butScrollIndex(1).Height
    butScrollTheme(1).Left = ScrollBar(1).Left

    butGoto.Top = lstTheme.Top
    butGoto.Height = butScrollTheme(1).Top + butScrollTheme(1).Height - butGoto.Top

    Filenr = FreeFile
    Open ProgramDir + datIndexINI For Input As Filenr
    ReDim IndexThemes(0)
    t$ = ""
    While Not EOF(Filenr)
```

```
        If Left$(t$, 1) = "[" Then
            t$ = Mid$(t$, 2, Len(t$) - 2) '[] wegschneiden
            lstIndex.AddItem t$
            i% = UBound(IndexThemes)
            lstIndex.ItemData(lstIndex.NewIndex) = i%
            While Not EOF(Filenr) And (Left$(t$, 1) <> "[")
                Line Input #Filenr, t$
                If Left$(t$, 1) <> "[" Then
                    t$ = Left$(t$, InStr(t$, "="))
                    IndexThemes(i%) = IndexThemes(i%) + t$
                End If
            Wend
            If Not EOF(Filenr) Then ReDim Preserve IndexThemes(i% + 1)
        Else
            Line Input #Filenr, t$
        End If
    Wend
    If lstIndex.ListCount > 0 Then lstIndex.ListIndex = 0
End Sub

Sub lstIndex_Click ()
    Dim t$, i%

    lstTheme.Clear
    t$ = IndexThemes(lstIndex.ItemData(lstIndex.ListIndex))
    i% = InStr(t$, "=")
    While i% <> 0
        lstTheme.AddItem Left$(t$, i% - 1)
        t$ = Mid$(t$, i% + 1, Len(t$))
        i% = InStr(t$, "=")
    Wend
    If lstTheme.ListCount > 0 Then lstTheme.ListIndex = 0
End Sub
```

frmUser - Objekte

```
Begin Form frmUser
  AutoRedraw    = -1 'True
  BorderStyle   = 0 'Keine
  ControlBox    = 0 'False
  MaxButton     = 0 'False
  MinButton     = 0 'False
  WindowState   = 2
  Begin SSPanel txtInfo
    BackColor     = &H00C0C0C0&
    BevelInner    = 1 'Inset
    BorderWidth   = 1
    ForeColor     = &H00000000&
  End
  Begin SSPanel butControl
    BackColor     = &H0000C0C0&
    BevelInner    = 2 'Raised
    BorderWidth   = 4
    Caption       = "Kommentar"
    FontBold      = 0 'False
    FontName      = "MS Sans Serif"
    FontSize      = 9.75
    Index         = 5
    RoundedCorners = 0 'False
  End
  Begin SSPanel butControl
    BackColor     = &H0000C0C0&
    BevelInner    = 2 'Raised
    BorderWidth   = 4
    Caption       = "Drucken"
    FontBold      = 0 'False
    FontName      = "MS Sans Serif"
    FontSize      = 9.75
    Index         = 4
    RoundedCorners = 0 'False
  End
  Begin SSPanel butControl
    BackColor     = &H0000C0C0&
    BevelInner    = 2 'Raised
    BorderWidth   = 4
    Caption       = "Suchen"
    FontBold      = 0 'False
    FontName      = "MS Sans Serif"
    FontSize      = 9.75
    Index         = 3
    RoundedCorners = 0 'False
  End
  Begin SSPanel butControl
    BackColor     = &H0000C0C0&
    BevelInner    = 2 'Raised
    BorderWidth   = 4
    Caption       = "Übersicht"
    FontBold      = 0 'False
    FontName      = "MS Sans Serif"
    FontSize      = 9.75
    Index         = 2
    RoundedCorners = 0 'False
  End
  Begin SSPanel butControl
    BackColor     = &H0000C0C0&
    BevelInner    = 2 'Raised
    BorderWidth   = 4
    Caption       = "Mehr"
    FontBold      = 0 'False
    FontName      = "MS Sans Serif"
    FontSize      = 9.75
    Index         = 1
    RoundedCorners = 0 'False
  End
  Begin SSPanel butControl
    BackColor     = &H0000C0C0&
    BevelInner    = 2 'Raised
    BorderWidth   = 4
    Caption       = "Zurück"
    FontBold      = 0 'False
    FontName      = "MS Sans Serif"
    FontSize      = 9.75
    Index         = 0
    RoundedCorners = 0 'False
  End
  Begin PictureBox animAusgabe
    AutoRedraw    = -1 'True
    BackColor     = &H00C0C0C0&
    BorderStyle   = 0 'Keine
    Visible       = 0 'False
  End
  Begin Timer ResetTimer
  End
  Begin Timer mciWAV_Timer
    Enabled       = 0 'False
  End
  Begin SSCommand butScroll
    BevelWidth    = 0
    Height        = 615
    Index         = 1
    Outline       = 0 'False
    Picture       = FRMUSER.FRX:0000
    RoundedCorners = 0 'False
    Visible       = 0 'False
  End
  Begin SSCommand butScroll
    BevelWidth    = 0
    Height        = 615
    Index         = 0
    Outline       = 0 'False
    Picture       = FRMUSER.FRX:0A7A
    RoundedCorners = 0 'False
    Visible       = 0 'False
  End
  Begin PictureBox picAusgabe
    AutoRedraw    = -1 'True
    BackColor     = &H00C0C0C0&
    BorderStyle   = 0 'Keine
    Visible       = 0 'False
  End
  Begin SSPanel frameAusgabe
    Alignment     = 8 'Center - BOTTOM
    BackColor     = &H00C0C0C0&
    BevelInner    = 1 'Inset
    BevelWidth    = 2
    Visible       = 0 'False
  End
  Begin MMControl mciAVI
    AutoEnable    = 0 'False
    BackVisible   = 0 'False
    DeviceType    = "AVIVideo"
    EjectVisible  = 0 'False
    Enabled       = 0 'False
    NextVisible   = 0 'False
    PauseVisible  = 0 'False
    PrevVisible   = 0 'False
    RecordVisible = 0 'False
    Silent        = -1 'True
    StepVisible   = 0 'False
    StopVisible   = 0 'False
    UpdateInterval = 0
    Visible       = 0 'False
  End
  Begin MMControl mciWAV
    AutoEnable    = 0 'False
    BackVisible   = 0 'False
    DeviceType    = "WaveAudio"
    EjectVisible  = 0 'False
    Enabled       = 0 'False
    NextVisible   = 0 'False
    PauseVisible  = 0 'False
    PrevVisible   = 0 'False
    RecordVisible = 0 'False
    StepVisible   = 0 'False
    StopVisible   = 0 'False
    UpdateInterval = 0
    Visible       = 0 'False
```

```
   End
   Begin PictureBox butTheme
      AutoRedraw     =  -1  'True
      Index          =  0
      Visible        =  0   'False
   End
   Begin SSCommand butHand
      Caption        =  "Linkshänder"
      Font3D         =  4  'Inset w/heavy shading
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  13.5
      ForeColor      =  &H00404080&
      Height         =  2055
      Index          =  1
      Outline        =  0   'False
      Picture        =  FRMUSER.FRX:14F4
      Visible        =  0   'False
      Width          =  2175
   End
   Begin SSCommand butHand
      Caption        =  "Rechtshänder"
      Font3D         =  4  'Inset w/heavy shading
      FontBold       =  -1  'True
      FontName       =  "MS Sans Serif"
      FontSize       =  13.5
      ForeColor      =  &H00404080&
      Height         =  2055
      Index          =  0
      Left           =  2520
      Outline        =  0   'False
      Picture        =  FRMUSER.FRX:3336
      Top            =  3480
      Visible        =  0   'False
   End
   Begin PictureBox txtAusgabe
      AutoRedraw     =  -1  'True
      BackColor      =  &H00E0FFFF&
      FillStyle      =  0  'Gefüllt
      FontTransparent =   0   'False
      ForeColor      =  &H00000000&
      Index          =  0
      Visible        =  0   'False
   End
   Begin PictureBox txtAusgabe
      AutoRedraw     =  -1  'True
      Index          =  1
      Visible        =  0   'False
   End
   Begin Shape ScrollBar
      BackColor      =  &H00004040&
      BackStyle      =  1  'Undurchsichtig
      Visible        =  0   'False
   End
End
```

frmUser - Deklarationen

```
'Konstante für butControl()
Const Zurueck = 0
Const mehr = 1
Const uebersicht = 2
Const Suchen = 3
Const drucken = 4
Const kommentar = 5
'Konstante für butHand()
Const rechts = 0
Const links = 1

'String-Konstante
Const strWait$ = "Warten Sie bitte einen Moment..."
Const strMore$ = "<Mehr> zeigt Ihnen weitere Themen"
Const strBack2$ = "<Zurück> zeigt Ihnen vorherige Themen"
Const strBack1$ = "<Zurück> zeigt Ihnen Ihr letztes Thema"
Const strSelect$ = "Wählen Sie bitte ein Thema"
Const strScroll$ = "Verschieben Sie den Text mit den Pfeiltasten"
Const strLoad$ = "Lade Daten zum Thema: "
Const strShow$ = "Derzeitige Auswahl: "
Const strHand$ = "Ordnen Sie die Schaltflächen gemäß Ihrer Gewöhnung an"
Const strInput$ = "Bitte machen Sie Ihre Eingabe, und beenden Sie mit <Schließen>"
Const strRequest$ = "Rückfrage! Wählen Sie bitte eine der Antworten"
Const strPrint01$ = "Die Daten werden ausgedruckt..."
Const strprint02$ = "Die Druckdaten werden erstellt..."


Dim minuten As Integer                  'Zähler für auto. Übersicht (Reset)
Dim distance As Integer                 'Distanze der Objekte zum Rand und zueinander
Dim BitsPixel As Integer                'Speicher die Farbtiefe der Bildausgabe
Dim TouchDown As Integer'Speicher für ThemeButton-Zustand
Dim wait As Integer                     '1. Verz. beim Scrollen
Dim EndCode As Integer                  'Touchspeicher für Programmabbruch-Code
Dim order As Integer                    'Buttons rechts(=0) oder links(=1)
Dim LoadedButtons As Integer            'Anzahl der geladenen Themebuttons
Dim actDecade As Integer                'sichtbare ThemeButton-Dekade
Dim SoundReplay As Integer              'Anzahl der noch abzuspeilenden Sounds
Dim SoundDelay As Long                  'Länge der Abspielpause in mSek
Dim SoundPlayed As Integer              'Sound wurde mind. 1x abgespielt
Dim WAVWhileAnim As Integer             'Sound nur solange Animation
Dim picWhileWAV As Integer              'Bild löschen nach Sound
Dim AnimReplay As Integer               'Anzahl der noch abzuspielenden Animationen
Dim FullPicSize As Integer 'Keine Textdatei erzeugt große Ausgabe
Dim picResize As Integer   'Scrollbar nur verändern, bei Resize

Dim bitmap As String 'Darzustellendes Bild nach Ablauf der Animationen
Dim TextFile As String   'Dargestellter Text

Dim Thema$    'Aktuell gewähltes Thema

Dim Theme() As ThemeRec                 'Informationen der geladenen Buttons

Dim way() As PathRec   'Speichert die gegangenen Pfade
Dim Goback As Integer  'Auswahl über Zurück-Button

Dim c&(3)     'Schattierungen für Buttons

Sub ButtonPressed ()
   EndCode = 0
   minuten = 0
   ResetTimer.Enabled = False
   ResetTimer.Enabled = True
   ResetTimer.Interval = 60000
End Sub

Sub ChangeDir (directory As String)
   Dim avi$, wav$, bmp$, arep%, wrep%, wdel%, waanim%, txt$
   Dim i%, printable%, Part$

   Part$ = "Action"
   If Dir$(directory, 16) = "" Then     'Prüft, ob Verzeichnis existiert
      directory = ""
      Exit Sub
   End If
```

```
    ChDir directory  'Verzeichnis  wechseln
    directory = CurDir$ 'Verzerzeichnis um gesamten PFadnamen ergänzen
    i% = (directory = StartDir)
    butHand(rechts).Visible = i%
    butHand(links).Visible = i%
    butControl(uebersicht).Visible = Not i%

    HideMedia 'MM stoppen und Ausgabebereiche unsichtbar
    HideButTheme  'Alte Buttons löschen
    SetBackgrPic 'Hintergrundbild laden
    SetbutTheme 'Neue Buttons laden


    If Not (Goback And way(UBound(way)).button <> -1) Then
    'MM_Dateien lesen und abspielen:
        wav$ = GetINFOString(Part$, "sound", "")
        avi$ = GetINFOString(Part$, "animation", "")
        bmp$ = GetINFOString(Part$, "picture", "")
        txt$ = GetINFOString(Part$, "text", "")
        arep% = Val(GetINFOString(Part$, "animreplay", "1"))
        wrep% = Val(GetINFOString(Part$, "soundreplay", "1"))
        wdel% = Val(GetINFOString(Part$, "sounddelay", "0"))
        waanim% = Val(GetINFOString(Part$, "sound_after_anim", "0")) <> 0
        picWhileWAV = Val(GetINFOString(Part$, "Picture_While_Sound", "0")) <> 0
        WAVWhileAnim = Val(GetINFOString(Part$, "Sound_While_Anim", "0"))
        printable% = Val(GetINFOString(Part$, "Print", "0")) <> 0
        PlayMedia wav$, wrep%, wdel%, waanim%, avi$, arep%, bmp$, txt$, printable%
    End If
    If i% Then PrintInfo strHand$
End Sub

Sub ClickDrucken ()
    Static puffer%
    Static PaperT
    Dim WMFprop As Single
    Dim w%, h%, bits%, i%
    Dim paperw, paperh, paperl
    Dim File As Integer, zeile$
    Dim ctrlFormat As RTF_Format
    Dim t$

    t$ = txtInfo.Caption

    If puffer% Then
        PrintInfo strRequest$
        i% = request(1, Str$(printer.Page) + " Seite(n) drucken", "Es liegen noch zu druckende Daten vor!" + Chr$(10) +
                    "Wollen Sie diese jetzt drucken oder neue hinzufügen", "Drucken|Hinzufügen")
        If i% = 1 Then
            PrintInfo strPrint01$
            If modDruck <> 0 Then printer.EndDoc
            PaperT = 0
        Else
            puffer% = False
        End If
    End If
    If puffer% = False Then
        PrintInfo strprint02$
        printer.ScaleMode = 3
        If ((modDruck And 2) <> 0) Then
            printer.Print " "
            printer.CurrentY = PaperT
            If PaperT > printer.ScaleHeight Then PaperT = 0
            If MedGetBMPSize(bitmap, w%, h%, bits%) = 0 Then
                paperw = (printer.ScaleWidth * 5 / 8) * w% / (5 / 8 * frmUser.ScaleWidth)
                paperh = paperw * (h% / w%)
                paperl = (printer.ScaleWidth - paperw) / 2
                MedSetBMP printer.hDC, Int(paperl), Int(PaperT), Int(paperw), Int(paperh), 3, bitmap
                PaperT = PaperT + paperh + 100
            Else
                If MedProofWMF(bitmap) = 0 Then
                    w% = picAusgabe.Width
                    h% = picAusgabe.Height
                    paperw = w% * (printer.ScaleWidth) / frmUser.ScaleWidth
                    paperh = paperw * (h% / w%)
                    paperl = (printer.ScaleWidth - paperw) / 2
                    MedStretchImg picAusgabe.hDC, 0, 0, w%, h%, printer.hDC, Int(paperl), Int(PaperT), Int(paperw), Int(paperh),
```

```
                                picAusgabe.hWnd
                    PaperT = PaperT + paperh + 100
                End If
            End If
        End If
        If ((modDruck And 4) <> 0) Then
            If (MedProofBMP(TextFile) = 0) Or (MedProofWMF(TextFile) = 0) Then
                'Bild von txtAusgabe(0) drucken
                printer.Print " "
                w% = txtAusgabe(0).Width
                h% = txtAusgabe(0).Height
                paperw = w% * (printer.ScaleWidth) / frmUser.ScaleWidth
                paperh = paperw * (h% / w%)
                paperl = 0
                MedStretchImg txtAusgabe(0).hDC, 0, 0, w%, h%, printer.hDC, Int(paperl), Int(PaperT), Int(paperw), Int(paperh),
                                txtAusgabe(0).hWnd
                PaperT = PaperT + paperh + 100
            Else
                If (Dir$(TextFile) <> "") And (TextFile <> "") Then
                    File = FreeFile
                    printer.FontName = fntInfoText
                    printer.FontSize = 10
                    printer.CurrentY = PaperT
                    Open TextFile For Input As File
                    Line Input #File, zeile$
                    If UCase$(Left$(zeile$, 5)) = "{\RTF" Then
                        Close File
                        ctrlFormat.FontSize = txtAusgabe(0).FontSize
                        ctrlFormat.ForeColor = 1
                        ReDim ColorTable(1)
                        ColorTable(0) = txtAusgabe(0).BackColor
                        ColorTable(1) = 0
                        ReDim tabs(1, 0)
                        tabs(0, 0) = -1
                        Open TextFile For Random As File Len = 1
                        GetRTFPart txtAusgabe(0), File, 2, ctrlFormat, True
                    Else
                        printer.Print zeile$
                        While Not EOF(File)
                            Line Input #File, zeile$
                            printer.Print
                        Wend
                    End If
                    Close File
                    PaperT = printer.CurrentY + 100
                End If
            End If
        End If
    'Druckauftrag beenden
        PrintInfo strRequest$
        i% = request(1, Str$(printer.Page) + " Seite(n) drucken", "Wann wollen Sie jetzt die erstellten Daten drucken?" + Chr$(10) +
                        "Um Daten später auszudrucken, betätigen Sie bitte nochmals die Taste <Drucken>", "Jetzt|Später")
        If i% = 1 Then
            PrintInfo strPrint01$
            If modDruck <> 0 Then printer.EndDoc
            PaperT = 0
            puffer% = False
        Else
            puffer% = True
        End If
    Else
        puffer% = False
    End If
    PrintInfo t$
End Sub

Sub ClickIndex ()
    Dim WAVTimer%, i%, t$, verz$, maxWay%
    Dim ButToPress%

    WAVTimer% = mciWAV_Timer.Enabled
    mciWAV_Timer.Enabled = False
    ResetTimer.Enabled = False
    mciWAV.Command = "Pause"
    mciAVI.Command = "Pause"

    On Error GoTo Ignore
```

```
    t$ = txtInfo.Caption
    PrintInfo strRequest$
    i% = request(2, "Thema suchen", "Geben Sie im Anschluß ein Thema ein," + Chr$(10) + "oder wählen Sie mit den Pfeiltasten eins aus
                den Listen." + Chr$(10) + "<Gehe zu> zeigt das Thema,       " + Chr$(10) + "<Schließen> bricht die Suche ab.", " Ok ")
    PrintInfo t$
    KeyboardMode = 4
    frmKeyboard.Tag = ""
    frmKeyboard.Top = Screen.Height - frmKeyboard.Height
    frmKeyboard.Visible = True
    frmIndex.Visible = True
    frmIndex.Tag = ""
    frmUser.Enabled = False
    While frmIndex.Tag = ""
        DoEvents
    Wend
    frmKeyboard.Visible = False
    frmIndex.Visible = False
    frmUser.Enabled = True
    frmUser.SetFocus
    frmUser.Refresh
    t$ = frmIndex.Tag
    If t$ <> "Cancel" Then
        i% = InStr(t$, ",")
        If i% = 0 Then
            If (Mid$(t$, 2, 1) <> ":") And (Left$(verz$, 1) <> "\") Then
                verz$ = StartDir + "\" + t$
            End If
            While Right$(verz$, 1) = "\"
                verz$ = Left$(verz$, Len(verz$) - 1)
            Wend
            If CurDir$ <> UCase$(verz$) Then
                ChangeDir verz
                maxWay% = UBound(way) + 1
                ReDim Preserve way(maxWay%)
                way(maxWay%).path = CurDir$
                way(maxWay%).button = -1
            End If
        Else
            verz$ = StartDir + "\" + Left$(t$, i% - 1)
            While Right$(verz$, 1) = "\"
                verz$ = Left$(verz$, Len(verz$) - 1)
            Wend
            If CurDir$ <> UCase$(verz$) Then ChangeDir verz
            t$ = UCase$("BUTTON" + Mid$(t$, i% + 1, Len(t$)))
            ButToPress% = -1
            For i% = 0 To LoadedButtons - 1
                If t$ = UCase$(Theme(i%).InitName) Then ButToPress% = i%
            Next i%
            If ButToPress% > -1 Then butTheme_Click ButToPress%
            maxWay% = UBound(way) + 1
            ReDim Preserve way(maxWay%)
            way(maxWay%).path = CurDir$
            way(maxWay%).button = ButToPress%
        End If
    End If
    If mciAVI.Mode = 529 Then mciAVI.Command = "Play"
    If mciWAV.Mode = 529 Then mciWAV.Command = "Play"
    mciWAV_Timer.Enabled  =  WAVTimer%
    ResetTimer.Enabled = True
SubExit:
    Exit Sub
Ignore:
    Resume SubExit
End Sub

Sub ClickKommentar ()
    Dim WAVTimer%, oldInfoText$, wunsch$
    Dim Filenr As Integer

    WAVTimer%  =  mciWAV_Timer.Enabled
    mciWAV_Timer.Enabled  = False
    ResetTimer.Enabled = False
    mciWAV.Command = "Pause"
    mciAVI.Command = "Pause"
    oldInfoText$ = txtInfo.Caption
    PrintInfo strRequest$
```

```
    If request(2, "Ihr Kommentar", "Sie können anschließend eingeben, welche Informationen oder Angebote Ihrer Meinung nach fehlen." +
                Chr$(10) + "Möchten Sie diese Möglichkeit nutzen?", "Ja|Nein") = 1 Then
        PrintInfo strInput$
        frmKeyboard.Tag = ""
        CenterForm frmKeyboard
        frmKeyboard.Show 1
        If frmKeyboard.Tag <> "" Then
            wunsch$ = Date$ + " - " + Trim$(frmKeyboard.Tag) + Chr$(13) + Chr$(10)
            Filenr = FreeFile
            Open ProgramDir + datPropose For Binary As Filenr
                Put Filenr, LOF(Filenr) + 1, wunsch$
            Close Filenr
        End If
    End If
    If mciAVI.Mode = 529 Then mciAVI.Command = "Play"
    If mciWAV.Mode = 529 Then mciWAV.Command = "Play"
    mciWAV_Timer.Enabled = WAVTimer%
    ResetTimer.Enabled = True
    PrintInfo oldInfoText$
End Sub

Sub ClickMehr ()
    Dim i%, butCount As Integer

    PrintInfo strBack2$
    For i% = actDecade * 10 To actDecade * 10 + 9
        butTheme(i%).Visible = False
    Next i%
    actDecade = actDecade + 1
    butCount = actDecade * 10 + 9
    If butCount > (LoadedButtons - 1) Then
        butCount = (LoadedButtons - 1)
        butControl(mehr).Visible = False
    End If
    For i% = actDecade * 10 To butCount
        butTheme(i%).Visible = True
    Next i%
    butControl(Zurueck).Visible = True
End Sub

Sub ClickZurueck ()
    Dim i%, butCount As Integer

    If actDecade > 0 Then
        butCount = actDecade * 10 + 9
        If butCount > (LoadedButtons - 1) Then butCount = (LoadedButtons - 1)
        For i% = actDecade * 10 To butCount
            butTheme(i%).Visible = False
        Next i%
        actDecade = actDecade - 1
        For i% = actDecade * 10 To actDecade * 10 + 9
            butTheme(i%).Visible = True
        Next i%
        If (actDecade = 0) Then
            Select Case UBound(way)
            Case 0: butControl(Zurueck).Visible = False
            Case 1: PrintInfo strSelect$
            Case Else: PrintInfo strBack1$
            End Select
        Else
            PrintInfo strMore$
        End If
        butControl(mehr).Visible = True
    Else
        i% = UBound(way) - 1
        ReDim Preserve way(i%)

        Goback = True
        If way(i%).path <> CurDir$ Then
            ChangeDir way(i%).path
        Else
            If (way(i%).button = -1) And (i% > 0) Then
                i% = i% - 1
                ReDim Preserve way(i%)
                ChangeDir way(i%).path
            End If
```

```
        End If
        If way(i%).button <> -1 Then butTheme_Click way(i%).button
        If i% = 0 Then butControl(Zurueck).Visible = False
        Goback = False
    End If
End Sub

Sub CreateTouch (Caption As String, bmp As String)
    Dim l%, t%, w%, h%, i%, lb%
    Dim col As Long, r%, g%, b%
    Dim TitleHeight As Integer
    Dim OldBKColor As Long
    Dim fs As Single, WMFprop As Single

    lb% = LoadedButtons - 1
    If lb% > 0 Then
        Load butTheme(lb%)
    End If
    butTheme(lb%).FontSize = fntButtonSize
    butTheme(lb%).Left = butControl(lb% And 1).Left
    butTheme(lb%).Top = distance + ((lb% Mod 10) \ 2) * (distance + butTheme(0).Height)

    LoadedButtons = LoadedButtons + 1

    TitleHeight = butTheme(lb%).TextHeight(Caption)
    If Caption = "" Then
        TitleHeight = -1
    End If
    l% = butTheme(lb%).ScaleLeft
    w% = butTheme(lb%).ScaleWidth
    fs = 0
    While (butTheme(lb%).TextWidth(Caption) > butTheme(lb%).Width - 4)
        If fs = butTheme(lb%).FontSize Then
            Caption = Left$(Caption, Len(Caption) - 1)
        Else
            fs = butTheme(lb%).FontSize
        End If
        butTheme(lb%).FontSize = butTheme(lb%).FontSize - 1
    Wend

        t% = 0
        h% = butTheme(lb%).ScaleHeight
        If Caption = "" Then Caption = " "
        For i% = 0 To 1
            butTheme(lb%).Line (i%, t% + i% + 1)-(w% - i% - 1, t% + i% + 1), c&(0)
            butTheme(lb%).Line (i%, t% + i% + 1)-(i%, t% + h% - i% + 1), c&(0)
            butTheme(lb%).Line (i% + 1, t% + h% - i% - 1)-(w% - 1, t% + h% - i% - 1), c&(2)
            butTheme(lb%).Line (w% - i% - 1, t% + i%)-(w% - i% - 1, t% + h% - 1), c&(2)
        Next i%
        butTheme(lb%).Line (l + 3, t% + 3)-(w% - 3, t% + h% - 3), c&(1), BF

    If bmp$ = "" Then
        butTheme(lb%).CurrentX = (w% - butTheme(lb%).TextWidth(Caption)) \ 2
        butTheme(lb%).CurrentY = (h% - butTheme(lb%).TextHeight(Caption)) \ 2
        butTheme(lb%).Print Caption
        Exit Sub
    End If
    t% = butTheme(lb%).ScaleTop + TitleHeight
    h% = butTheme(lb%).ScaleHeight - (t% - butTheme(lb%).ScaleTop)
    butTheme(lb%).Line (0, t%)-(w%, t% + h%), c&(1), BF
    butTheme(lb%).Line (0, t%)-(w%, t%), 0
    butTheme(lb%).Line (0, t% + 1)-(w%, t% + 1), c&(0)
    butTheme(lb%).Line (0, t% + 1)-(0, t% + h% - 1), c&(0)
    butTheme(lb%).Line (1, t% + h% - 1)-(w% - 1, t% + h% - 1), c&(2)
    butTheme(lb%).Line (w% - 1, t% + 1)-(w% - 1, t% + h% - 1), c&(2)

    butTheme(lb%).CurrentX = (w% - butTheme(lb%).TextWidth(Caption)) \ 2
    butTheme(lb%).CurrentY = (TitleHeight% - butTheme(lb%).TextHeight(Caption)) \ 2
    butTheme(lb%).Print Caption

    For i% = 5 To 6
        butTheme(lb%).Line (i%, t% + i%)-(w% - i%, t% + i%), c&(2)
        butTheme(lb%).Line -(w% - i%, t% + h% - i%), c&(0)
        butTheme(lb%).Line -(i%, t% + h% - i%), c&(0)
        butTheme(lb%).Line -(i%, t% + i%), c&(2)
    Next i%
```

```
    butTheme(lb%).Line (0, t% + 1)-(w%, t% + 1), c&(0)
    butTheme(lb%).Line (0, t% + 1)-(0, t% + h% - 1), c&(0)
    butTheme(lb%).Line (1, t% + h% - 1)-(w% - 1, t% + h% - 1), c&(2)
    butTheme(lb%).Line (w% - 1, t% + 1)-(w% - 1, t% + h% - 1), c&(2)
    butTheme(lb%).Line (l% + 7, t% + 7)-(w% - 7, t% + h% - 7), c&(3), BF
    butTheme(lb%).Refresh
    If MedGetWMFProp(bmp$, WMFprop) = 0 Then
        OldBKColor = txtAusgabe(0).BackColor
        txtAusgabe(0).BackColor = c&(3)
        If WMFprop < (h% / w%) Then
            txtAusgabe(0).Width = w% - 13
            txtAusgabe(0).Height = txtAusgabe(0).Width * WMFprop
        Else
            txtAusgabe(0).Height = h% - 13
            txtAusgabe(0).Width = txtAusgabe(0).Height / WMFprop
        End If
        txtAusgabe(0).Picture = LoadPicture(bmp$)
        If WMFprop < (h% / w%) Then
            MedCopyImg txtAusgabe(0).hDC, 0, 0, txtAusgabe(0).Width, txtAusgabe(0).Height, butTheme(lb%).hDC, l% + 7,
                        t% + 7 + (h% - 7 - txtAusgabe(0).Height) / 2
        Else
            MedCopyImg txtAusgabe(0).hDC, 0, 0, txtAusgabe(0).Width, txtAusgabe(0).Height, butTheme(lb%).hDC,
                        (butTheme(lb%).Width - txtAusgabe(0).Width) / 2, t% + 7
        End If
        butTheme(lb%).Refresh
        txtAusgabe(0).Picture = LoadPicture("")
        txtAusgabe(0).BackColor = OldBKColor
        txtAusgabe(0).Width = txtAusgabe(1).Width
    Else
        MedSetBMP butTheme(lb%).hDC, l% + 7, t% + 7, w% - 13, h% - 13, modThemePic, bmp
    End If
End Sub


Function GetINFOString$ (Theme$, KeyName$, Default$)
    Dim anz%, RetString$, verz$

    RetString$ = Space$(50)
    If Right$(CurDir$, 1) <> "\" Then
        verz$ = CurDir$ + "\"
    Else
        verz$ = CurDir$
    End If
    anz% = GetPrivateProfileString(Theme$, KeyName$, Default$, RetString$, 50, verz$ + datInfoIni)
    GetINFOString$ = Left$(RetString$, anz%)
End Function

Sub HideButTheme ()
    'Alle butTheme(#>0) löschen
    While LoadedButtons > 1
        LoadedButtons = LoadedButtons - 1
        Unload butTheme(LoadedButtons)
    Wend
    butTheme(0).Visible = False
    butControl(mehr).Visible = False
End Sub

Sub HideMedia ()
    Dim i%

    exitDelay = True
    picResize = False
    SoundReplay = 0
    mciWAV_Timer.Interval = 0
    mciWAV_Timer.Enabled = False
    mciAVI.Command = "Close"
    mciWAV.Command = "Close"
    picAusgabe.Visible = False
    animAusgabe.Visible = False
    If (BitsPixel <= 8) Then picAusgabe.Picture = LoadPicture("")
    frameAusgabe.Visible = False
    txtAusgabe(1).Visible = False
    VisibleScroll False
    txtAusgabe(0).Cls
    txtAusgabe(1).Cls
    picAusgabe.Cls
```

```
    i% = SetPicSize(picAusgabe, 0, 0)
End Sub

Sub PlayMedia (wav$, wrep%, wdel%, waanim%, avi$, arep%, bmp$, txt$, printable%)
    Dim w%, h%, fullheight%, anim%
    Dim bits%, prop As Single
    Dim Filenr As Integer
    Dim rtf As String * 5
    Dim zeile$
    Dim ctrlFormat As RTF_Format

    frmUser.Refresh
    AnimReplay = arep%
    SoundReplay = wrep%
    SoundDelay = 1000 * Abs(wdel%)
    mciWAV_Timer.Enabled = False
    If modTon <> 0 Then mciWAV_Timer.Interval = SoundDelay
    FullPicSize = ((txt$ = "") And (CurDir$ <> StartDir))

    HeightOfText = 0
    StartCopyY = 0

    bitmap = bmp$
    TextFile = txt$

    SoundPlayed = (wav$ = "")

    If txtAusgabe(0).Height < 1000 Then txtAusgabe(0).Height = 1000
    txtAusgabe(0).Picture = LoadPicture("")
    If (Dir$(txt$) <> "") And (txt$ <> "") Then
        Filenr = FreeFile
        Open txt$ For Input As Filenr
            Line Input #Filenr, rtf
        Close Filenr
        If UCase$(rtf) = "{\RTF" Then
'Text-Datei im Rich-Text-Format
            ctrlFormat.FontSize = txtAusgabe(0).FontSize
            ctrlFormat.ForeColor = 1

            ReDim ColorTable(1)
            ColorTable(0) = txtAusgabe(0).BackColor
            ColorTable(1) = 0
            ReDim tabs(1, 0)
            tabs(0, 0) = -1

            Open txt$ For Random As Filenr Len = 1
                GetRTFPart txtAusgabe(0), Filenr, 2, ctrlFormat, False
            Close Filenr
        Else
            If MedGetBMPSize(txt$, w%, h%, bits%) = 0 Then
'Text-Datei im BMP-Format
                'Ausgabehöhe an Bildhöhe anpassen
                HeightOfText = (txtAusgabe(0).Width / w%) * h%
                txtAusgabe(0).Height = HeightOfText
                'Palette setzen
                If (BitsPixel <= 8) And (bits% <= 8) Then txtAusgabe(0).Picture = LoadPicture(txt$)
                'Bild laden
                MedSetBMP txtAusgabe(0).hDC, 0, 0, txtAusgabe(0).Width, HeightOfText, 3, txt$
            Else
                If MedGetWMFProp(txt$, prop) = 0 Then
'Text-Datei im WMF-Format
                    'Ausgabehöhe an Bildhöhe anpassen
                    HeightOfText = txtAusgabe(0).Width * prop
                    txtAusgabe(0).Height = HeightOfText
                    'WMF laden
                    txtAusgabe(0).Picture = LoadPicture(txt$)
                Else
'Text-Datei im ANSI-Format
                    'Standartformat setzen
                    ctrlFormat.FontSize = 12
                    ctrlFormat.FontBold = False
                    ctrlFormat.FontItalic = False
                    ctrlFormat.FontStrikethru = False
                    ctrlFormat.FontUnderline = False
                    ReDim ColorTable(1)
                    ColorTable(0) = txtAusgabe(1).BackColor
```

```
                    ColorTable(1) = 0
                    ctrlFormat.ForeColor = 1
                    ctrlFormat.BackColor = 0
                    ctrlFormat.Align = 0
                    ctrlFormat.FirstLeftMargin = 0
                    ctrlFormat.LeftMargin = 0
                    ctrlFormat.RightMargin = 0
                    'Zeile lesen und ausgeben
                    Open txt$ For Input As Filenr
                    While Not EOF(Filenr)
                        Line Input #Filenr, zeile$
                        zeile$ = zeile$ + Chr$(13)
                        NeuerAbsatz = True
                        Format_Output txtAusgabe(0), zeile$, ctrlFormat
                    Wend
                    Close Filenr
                End If
            End If
            txtAusgabe(0).Height = HeightOfText
        End If
    Else
        txt$ = ""
    End If

    anim% = (MedGetAVISize(avi$, w%, h%) = 0) And AnimReplay <> 0
    If anim% Then
        If SetPicSize(animAusgabe, w%, h%) Then
            mciAVI.FileName = avi$
            mciAVI.Command = "Open"
            frameAusgabe.Visible = True
            frameAusgabe.Refresh
            animAusgabe.Visible = True
            animAusgabe.Refresh
            mciAVI_Done 1
        Else
            anim% = False
        End If
    End If
    If (modTon <> 0) And (MedProofWAV(wav$) = 0) And (SoundReplay <> 0) Then
        mciWAV.FileName = wav$
        mciWAV.Command = "Open"
        If (anim% And Not waanim%) Or Not anim% Then mciWAV.Command = "Play"
    End If
    If Not anim% Then SetPic bitmap
    If txt$ <> "" Then
        If HeightOfText < txtAusgabe(1).Height Then
            VisibleScroll False
            txtAusgabe(1).Height = HeightOfText
        End If
        MedCopyImg txtAusgabe(0).hDC, 0, StartCopyY, txtAusgabe(1).Width, txtAusgabe(1).Height, txtAusgabe(1).hDC, 0, 0
        txtAusgabe(1).Visible = True
        frmUser.Refresh
        If HeightOfText > txtAusgabe(1).Height Then VisibleScroll True
    End If
    picResize = True
    butControl(drucken).Visible = printable% And (modDruck > 1)
End Sub

Sub PrintInfo (Caption$)
    Dim fs
    frmUser.FontSize = fntButtonSize
    fs = 0
    While (frmUser.TextWidth(Caption$) > txtInfo.Width - 4) And (fs <> frmUser.FontSize)
        fs = frmUser.FontSize
        frmUser.FontSize = frmUser.FontSize - 1
    Wend
    txtInfo.FontSize = frmUser.FontSize
    txtInfo.Caption = Caption$
End Sub

Sub SetBackgrPic ()
    Static actBackgr$
    Dim i%, w%, h%, x%, y%, bits%
    Dim verz$

    If Right$(CurDir$, 1) <> "\" Then
```

```
        verz$ = CurDir$ + "\"
    Else
        verz$ = CurDir$
    End If
    'Backgr.bmp in vorangehenden Verzeichnissen suchen
    If Dir$(verz$ + datBckBMP) = "" Then
    While (Dir$(verz$ + datBckBMP) = "") And (Len(verz$) > 3) And (UCase$(verz$) <> StartDir + "\")
        verz$ = Left$(verz$, Len(verz$) - 1)
        While (Len(verz$) > 2) And (Right$(verz$, 1) <> "\")
            verz$ = Left$(verz$, Len(verz$) - 1)
        Wend
    Wend
    'Es gibt keinen Hintergrund
    If Dir$(verz$ + datBckBMP) = "" Then
        actBackgr$ = ""
        frmUser.Cls
        Exit Sub
    End If
    End If
    'Hintergrund wird nicht verändert
    If actBackgr$ = verz$ + datBckBMP Then Exit Sub
    'Hintergrund wird verändert
    actBackgr$ = verz$ + datBckBMP
    frmUser.Cls
    If MedGetBMPSize(verz$ + datBckBMP, w%, h%, bits%) <> 0 Then Exit Sub
    Select Case modHintergrund
    Case 0: MedSetBMP frmUser.hDC, 0, 0, frmUser.ScaleWidth, frmUser.ScaleHeight, 1, verz$ + datBckBMP
    Case 1: MedSetBMP frmUser.hDC, 0, 0, frmUser.ScaleWidth, frmUser.ScaleHeight, 2, verz$ + datBckBMP
    Case 2
        MedSetBMP frmUser.hDC, 0, 0, frmUser.ScaleWidth, frmUser.ScaleHeight, 0, verz$ + datBckBMP
        y% = 0
        x% = w%
        Do
            While x% < frmUser.ScaleWidth
                MedMoveImg frmUser.hDC, 0, 0, w%, h%, x%, y%
                x% = x% + w%
            Wend
            y% = y% + h%
            x% = 0
        Loop Until y% >= frmUser.ScaleHeight
    End Select
    frmUser.Refresh
End Sub

Sub SetbutTheme ()
    Dim File As Integer
    Dim butCount As Integer
    Dim zeile$, bmp$, i%
    butCount = 0

'Prüfen ob INFO_INI vorhanden
    If Dir$(datInfoIni) = "" Then Exit Sub
'Alle Button-Themenamen lesen
    File = FreeFile
    Open datInfoIni For Input As File
    While Not EOF(File)
        Line Input #File, zeile$
        If UCase$(Left$(zeile$, 7)) = "[BUTTON" Then
            ReDim Preserve Theme(butCount)
            zeile$ = Mid$(zeile$, 2, Len(zeile$) - 2) '[] wegschneiden
            Theme(butCount).InitName = zeile$
            butCount = butCount + 1
        End If
    Wend
    Close File
'Keine Buttons zu erzeugen
    If butCount = 0 Then Exit Sub
    PrintInfo strWait$
    txtInfo.Refresh
'Lese Info-Daten

    For i% = 0 To butCount - 1
        Theme(i%).datPic = GetINFOString(Theme(i%).InitName, "picture", "")
        Theme(i%).datAnim = GetINFOString(Theme(i%).InitName, "animatioin", "")
        Theme(i%).datSound = GetINFOString(Theme(i%).InitName, "sound", "")
        Theme(i%).datText = GetINFOString(Theme(i%).InitName, "text", "")
```

```
        zeile$ = GetINFOString(Theme(i%).InitName, "dir", "")
'ggf. Backslash am Ende löschen
        If Left$(zeile$, 1) = "\" Then zeile$ = Right$(zeile$, Len(zeile$) - 1)
        Theme(i%).dir = zeile$
        Theme(i%).ItemNr = GetINFOString(Theme(i%).InitName, "ItemNr", "")
        Theme(i%).AnimReplay = Val(GetINFOString(Theme(i%).InitName, "animreplay", "1"))
        Theme(i%).SoundReplay = Val(GetINFOString(Theme(i%).InitName, "soundreplay", "1"))
        Theme(i%).printable = Val(GetINFOString(Theme(i%).InitName, "print", "0")) <> 0
        Theme(i%).title = GetINFOString(Theme(i%).InitName, "title", "")
        bmp$ = GetINFOString(Theme(i%).InitName, "thumbnail", "")
        If bmp$ = "->" Then bmp$ = Theme(i%).datPic
'ggf. [->] durch unter DIR ang. Verzeichnis ersetzen
        If Left$(bmp$, 2) = "->" Then
            bmp$ = Right$(bmp$, Len(bmp$) - 2)   '-> löschen
            If Left$(bmp$, 1) = "\" Then
                bmp$ = Theme(i%).dir + bmp$       'Verzeichnis einsetzen
            Else
                bmp$ = Theme(i%).dir + "\" + bmp$ 'ggf. Backslash hinzufügen
            End If
        End If
        If (MedProofBMP(bmp$) <> 0) And (MedProofWMF(bmp$) <> 0) Then bmp$ = ""
        CreateTouch Theme(i%).title, bmp$
    Next i%
    LoadedButtons = LoadedButtons - 1
    If butCount > 10 Then
        PrintInfo strMore$
    Else
        PrintInfo strSelect$
    End If
    If butCount > 10 Then
        butCount = 10
        butControl(mehr).Visible = True
    Else
        butControl(mehr).Visible = False
    End If
    actDecade = 0
    For i% = 0 To butCount - 1
        butTheme(i%).Visible = True
    Next i%
End Sub

Sub SetPic (bmp$)
    Dim w%, h%, bits%, dw%, dh%, dummy%
    Dim WMFprop As Single

    If MedGetBMPSize(bmp$, w%, h%, bits%) = 0 Then 'Bild ist eine BMP-Datei
        dw% = w%: dh% = h%
        If SetPicSize(picAusgabe, dw%, dh%) Then      'Bild paßt so wie es ist
            If (bits% <= 8) And (BitsPixel <= 8) Then
                picAusgabe.Picture = LoadPicture(bmp$)
            Else
                If (BitsPixel <= 8) Then picAusgabe.Picture = LoadPicture("")
                MedSetBMP picAusgabe.hDC, 0, 0, w%, h%, 0, bmp$                'Bild normal ausgeben
            End If
        Else
            If (w% - dw%) > (h% - dh%) Then
                dh% = dw% * (h% / w%)                 'Bild einpassen, Höhe anpassen
            Else
                dw% = dh% * (w% / h%)                 'Bild einpassen, Breite anpassen
            End If
            dummy% = SetPicSize(picAusgabe, dw%, dh%)
            If (BitsPixel <= 8) Then
                If bits% <= 8 Then
                    picAusgabe.Picture = LoadPicture(bmp$)
                Else
                    picAusgabe.Picture = LoadPicture("")
                End If
            End If
            MedSetBMP picAusgabe.hDC, 0, 0, dw%, dh%, 3, bmp$
        End If
        frameAusgabe.Visible = True
        frameAusgabe.Refresh
        picAusgabe.Visible = True
        picAusgabe.Refresh
    Else
        If MedGetWMFProp(bmp$, WMFprop) = 0 Then
```

```
            If WMFprop < (4 / 5) Then
                w% = frmUser.ScaleWidth * (5 / 8)
                h% = w% * WMFprop
            Else
                h% = frmUser.ScaleHeight \ 2
                w% = h% / WMFprop
            End If
            dummy% = SetPicSize(picAusgabe, w%, h%)
            picAusgabe.Picture = LoadPicture(bmp$)
            frameAusgabe.Visible = True
            frameAusgabe.Refresh
            picAusgabe.Visible = True
            picAusgabe.Refresh
        Else
            picAusgabe.Visible = False
            frameAusgabe.Visible = False
            dummy% = SetPicSize(picAusgabe, 0, 0)
        End If
    End If
End Sub

Function SetPicSize (Ausgabe As Control, w%, h%) As Integer
    Dim wPlatz%, hPlatz%
    Dim vis%

    vis% = Ausgabe.Visible
    Ausgabe.Visible = False
    frameAusgabe.Visible = False

    wPlatz% = frmUser.ScaleWidth * (5 / 8)
    If FullPicSize Then
        hPlatz% = frmUser.ScaleHeight - 16 - 3 * distance - txtInfo.Height
    Else
        hPlatz% = frmUser.ScaleHeight \ 2
    End If
    SetPicSize = True
    If w% > wPlatz% Then
        w% = wPlatz%
        SetPicSize = False
    End If
    If h% > hPlatz% Then
        h% = hPlatz%
        SetPicSize = False
    End If
    Ausgabe.Width = w%
    Ausgabe.Height = h%
    If order = 0 Then
        Ausgabe.Left = distance + 8 + (wPlatz% - w%) \ 2
    Else
        Ausgabe.Left = frmUser.ScaleWidth - distance - 8 - wPlatz% + (wPlatz% - w%) \ 2
    End If
    If FullPicSize Then
        Ausgabe.Top = distance + 8 + (hPlatz% - h%) \ 2
    Else
        Ausgabe.Top = distance + 8
    End If
    frameAusgabe.Width = Ausgabe.Width + 16
    frameAusgabe.Height = Ausgabe.Height + 16
    frameAusgabe.Left = Ausgabe.Left - 8
    frameAusgabe.Top = Ausgabe.Top - 8
    frameAusgabe.Visible = vis%
    frameAusgabe.Refresh
    Ausgabe.Visible = vis%
    VisibleScroll False
    If Not FullPicSize Then
        txtAusgabe(1).Top = frameAusgabe.Top + frameAusgabe.Height + distance
        txtAusgabe(1).Height = txtInfo.Top - txtAusgabe(1).Top - distance
        If HeightOfText > 0 Then
            If HeightOfText < txtAusgabe(1).Height Then
                txtAusgabe(1).Height = HeightOfText
                VisibleScroll False
            End If
            StartCopyY = 0
            MedCopyImg txtAusgabe(0).hDC, 0, StartCopyY, txtAusgabe(1).Width, txtAusgabe(1).Height, txtAusgabe(1).hDC, 0, 0
            txtAusgabe(1).Refresh
            If picResize And (HeightOfText > txtAusgabe(1).Height) Then VisibleScroll True
```

```
        End If
    End If
    butScroll(0).Top = txtAusgabe(1).Top
    butScroll(1).Top = txtAusgabe(1).Top + txtAusgabe(1).Height - butScroll(1).Height
    Scrollbar.Top = butScroll(0).Top
    Scrollbar.Height = butScroll(1).Top + butScroll(1).Height - Scrollbar.Top
End Function

Sub VisibleScroll (Mode%)
    Dim dtxt%

    If Mode% Then
        PrintInfo strScroll$
        dtxt% = distance
    Else
        dtxt% = distance + (butScroll(0).Width / 2)
        If txtInfo.Caption = strScroll$ Then
            If Thema$ <> "" Then
                PrintInfo strShow$ + Thema$
            Else
            PrintInfo strSelect$
            End If
        End If
    End If
    Select Case order
    Case 0: txtAusgabe(1).Left = dtxt%
    Case 1: txtAusgabe(1).Left = frmUser.ScaleWidth - dtxt% - txtAusgabe(1).Width
    End Select
    butScroll(0).Visible = Mode%
    butScroll(1).Visible = Mode%
    Scrollbar.Visible = Mode%
    frmUser.Refresh
    TouchDown = False
End Sub
```

frmUser - Ereignisse

```
Sub butControl_Click (index As Integer)
    Dim i%

    ButtonPressed
    Select Case index
    Case 0: ClickZurueck
    Case 1: ClickMehr
    Case 2
        i% = UBound(way) + 1
        ReDim Preserve way(i%)
        way(i%) = way(i% - 1)
        ChangeDir StartDir
    Case 3: ClickIndex
    Case 4: ClickDrucken
    Case 5: ClickKommentar
    End Select
End Sub

Sub butControl_DblClick (index As Integer)
    If index = 2 Then butControl_Click index
End Sub

Sub butControl_MouseDown (index As Integer, button As Integer, Shift As Integer, x As Single, y As Single)
    If Not TouchDown Then
        butControl(index).BevelInner = 1
        butControl(index).BevelOuter = 1
        butControl(index).Move butControl(index).Left + 2, butControl(index).Top + 2
        TouchDown = True
    End If
End Sub

Sub butControl_MouseUp (index As Integer, button As Integer, Shift As Integer, x As Single, y As Single)
    If TouchDown Then
        butControl(index).BevelInner = 2
        butControl(index).BevelOuter = 2
        butControl(index).Move butControl(index).Left - 2, butControl(index).Top - 2
        TouchDown = False
    End If
End Sub
```

```
Sub butEnde_Click ()
    mouseVisible True
    Unload frmUser
    End
End Sub

Sub butHand_Click (index As Integer)
    Dim i%, w%, h%
    ReDim vis(4) As Integer

    If index = order Then
        'Ende-Tastkombination ergänzen
        'Prinzip:butÜbersicht-Form-Form-butÜbersicht-Form-butÜbersicht-butÜbersicht
        Select Case EndCode
        Case 0: EndCode = 1
        Case 7: EndCode = &HF
        Case &H1F: EndCode = &H3F
        Case &H3F
            Beep
            If request(1, "KIOSK von Percy Wippler", "Sie beenden das KIOSK-Programm", "Ok|Abbrechen") = 1 Then
                mouseVisible True
                Unload frmUser
                End
            Else
                EndCode = 0
            End If
        Case Else: EndCode = 0
        End Select
        Exit Sub
    Else
        EndCode = 0
    End If
    order = index
    vis(0) = Scrollbar.Visible
    Scrollbar.Visible = False
    For i% = 0 To 1
        butScroll(i%).Visible = False
        butHand(i%).Visible = False
    Next i%
    vis(1) = frameAusgabe.Visible
    vis(2) = picAusgabe.Visible
    vis(3) = animAusgabe.Visible
    picAusgabe.Visible = False
    animAusgabe.Visible = False
    frameAusgabe.Visible = False
    vis(4) = txtAusgabe(1).Visible
    txtAusgabe(1).Visible = False
    txtInfo.Visible = False

    Select Case order
    Case 0
        For i% = 0 To LoadedButtons - 1
            butTheme(i%).Left = frmUser.ScaleWidth - (2 - (i% Mod 2)) * (butTheme(i%).Width + distance)
        Next i%
        For i% = 0 To 5
            butControl(i%).Left = frmUser.ScaleWidth - (2 - (i% Mod 2)) * (butControl(i%).Width + distance)
        Next i%
        txtInfo.Left = distance
        txtAusgabe(1).Left = distance - (Not butScroll(0).Visible) * butScroll(0).Width / 2
        butScroll(0).Left = distance + txtAusgabe(1).Width
        butScroll(1).Left = butScroll(0).Left
        Scrollbar.Left = butScroll(0).Left
        If vis(3) Then
            w% = animAusgabe.Width
            h% = animAusgabe.Height
            i% = SetPicSize(animAusgabe, w%, h%)
        Else
            w% = picAusgabe.Width
            h% = picAusgabe.Height
            i% = SetPicSize(picAusgabe, w%, h%)
        End If
        w% = frmUser.ScaleWidth * 5 / 8
        butHand(links).Left = distance + (w% - 2 * butHand(links).Width - distance) \ 2
        butHand(rechts).Left = butHand(links).Left + butHand(links).Width + distance
    Case 1
```

```
        For i% = 0 To LoadedButtons - 1
            butTheme(i%).Left = distance + (i% And 1) * (distance + butTheme(i%).Width)
        Next i%
        For i% = 0 To 5
            butControl(i%).Left = distance + (i% And 1) * (distance + butControl(i%).Width)
        Next i%
        txtInfo.Left = frmUser.ScaleWidth - distance - txtInfo.Width
        txtAusgabe(1).Left = frmUser.ScaleWidth - (distance - (Not butScroll(0).Visible) * butScroll(0).Width / 2) - txtAusgabe(1).Width
        butScroll(0).Left = frmUser.ScaleWidth - distance - txtAusgabe(1).Width - butScroll(0).Width
        butScroll(1).Left = butScroll(0).Left
        Scrollbar.Left = butScroll(0).Left
        If vis(3) Then
            w% = animAusgabe.Width
            h% = animAusgabe.Height
            i% = SetPicSize(animAusgabe, w%, h%)
        Else
            w% = picAusgabe.Width
            h% = picAusgabe.Height
            i% = SetPicSize(picAusgabe, w%, h%)
        End If
        w% = frmUser.ScaleWidth * 5 / 8
        butHand(links).Left = frmUser.ScaleWidth - distance - (w% - distance) \ 2 - butHand(links).Width
        butHand(rechts).Left = butHand(links).Left + butHand(links).Width + distance
    End Select
    Scrollbar.Visible = vis(0)
    frameAusgabe.Visible = vis(1)
    picAusgabe.Visible = vis(2)
    animAusgabe.Visible = vis(3)
    txtAusgabe(1).Visible = vis(4)
    txtInfo.Visible = True
    For i% = 0 To 1
        butScroll(i%).Visible = vis(0)
        butHand(i%).Visible = True
    Next i%
    PrintInfo strSelect$
End Sub

Sub butScroll_MouseDown (index As Integer, button As Integer, Shift As Integer, x As Single, y As Single)
    Static ScrollHeight As Single

    ScrollHeight = .6
    ButtonPressed
    If Not TouchDown Then
        TouchDown = True
        On Error GoTo fehler
        While  TouchDown
            Select Case index
            Case 0
                If StartCopyY >= ScrollHeight Then
                    StartCopyY = StartCopyY - ScrollHeight
                Else
                    StartCopyY = 0
                End If
            Case 1
                If StartCopyY <= HeightOfText - txtAusgabe(1).Height - ScrollHeight Then
                    StartCopyY = StartCopyY + ScrollHeight
                Else
                    StartCopyY = HeightOfText - txtAusgabe(1).Height
                End If
            End Select
            MedCopyImg txtAusgabe(0).hDC, 0, StartCopyY, txtAusgabe(1).Width, txtAusgabe(1).Height, txtAusgabe(1).hDC, 0, 0
            txtAusgabe(1).Refresh
            DoEvents
            ScrollHeight = ScrollHeight + .3
        Wend
    End If
    Exit Sub
fehler:
    Resume Next
    Exit Sub
End Sub

Sub butScroll_MouseUp (index As Integer, button As Integer, Shift As Integer, x As Single, y As Single)
    If TouchDown Then
        TouchDown = False
    End If
```

```
    exitDelay = True
    wait = 0
End Sub

Sub butTheme_Click (index As Integer)
    Dim r As ThemeRec
    Dim wav$, avi$, bmp$, txt$
    Dim wrep%, wdel%, arep%, waanim%
    Dim n%, i%, printable%

    ButtonPressed
    If Not Goback And ((way(UBound(way)).button <> index) Or (way(UBound(way)).path <> CurDir$)) Then
        n% = UBound(way)
        If n% = 30 Then
            For i% = 1 To n% - 1
                way(i%) = way(i% + 1)
            Next i%
            way(0).path = StartDir
            way(0).button = -1
        Else
            ReDim Preserve way(n% + 1)
        End If
        butControl(Zurueck).Visible = True
    End If

    r = Theme(index)
    Thema$ = r.title
    PrintInfo strLoad$ + Thema$
    If r.dir <> "" Then
        ChangeDir r.dir
        If Not Goback Then
            way(UBound(way)).path = CurDir$
            way(UBound(way)).button = -1
        End If
    Else
        If Not Goback Then
            way(UBound(way)).path = CurDir$
            way(UBound(way)).button = index
        End If
        'MM-Ausgabe löschen
        HideMedia
        'MM_Dateien lesen und abspielen:
        wav$ = GetINFOString(r.InitName, "sound", "")
        avi$ = GetINFOString(r.InitName, "animation", "")
        bmp$ = GetINFOString(r.InitName, "picture", "")
        txt$ = GetINFOString(r.InitName, "text", "")
        arep% = Val(GetINFOString(r.InitName, "animreplay", "1"))
        wrep% = Val(GetINFOString(r.InitName, "soundreplay", "1"))
        wdel% = Val(GetINFOString(r.InitName, "sounddelay", "0"))
        waanim% = Val(GetINFOString(r.InitName, "sound_after_anim", "0")) <> 0
        WAVWhileAnim = Val(GetINFOString(r.InitName, "Sound_While_Anim", "0"))
        picWhileWAV = Val(GetINFOString(r.InitName, "Picture_While_Sound", "0")) <> 0
        printable% = Val(GetINFOString(r.InitName, "Print", "0")) <> 0
        PlayMedia wav$, wrep%, wdel%, waanim%, avi$, arep%, bmp$, txt$, printable%
    End If
    If (r.dir = "") And Not (butScroll(0).Visible) Then
        If (Trim$(r.title) <> "") Then
            PrintInfo strShow$ + Thema$
        Else
            PrintInfo strSelect$
        End If
    End If
End Sub

Sub butTheme_MouseDown (index As Integer, button As Integer, Shift As Integer, x As Single, y As Single)
    If Not TouchDown Then
        butTheme(index).Move butTheme(index).Left + 5, butTheme(index).Top + 5
        TouchDown = True
    End If
End Sub

Sub butTheme_MouseUp (index As Integer, button As Integer, Shift As Integer, x As Single, y As Single)
    If TouchDown Then
        butTheme(index).Move butTheme(index).Left - 5, butTheme(index).Top - 5
        TouchDown = False
    End If
End Sub
```

```
Sub Form_Click ()
    'Ende-Tastkombination ergänzen
    'Prinzip:butÜbersicht-Form-Form-butÜbersicht-Form-butÜbersicht-butÜbersicht
    Select Case EndCode
    Case 1: EndCode = 3
    Case 3: EndCode = 7
    Case &HF: EndCode = &H1F
    Case Else: EndCode = 0
    End Select
End Sub

Sub Form_DblClick ()
    Form_Click
End Sub

Sub Form_Load ()
    Dim dw As Single
    Dim dh As Single
    Dim i%, w%, r%, g%, b%, fs

    mouseVisible (-modMaus)
    KeyboardMode = 2
    BitsPixel = MedGetDCColors(picAusgabe.hDC)
    ChDrive Left$(StartDir, 2)
    ChDir StartDir

    ReDim way(0)
    way(0).path = StartDir
    way(0).button = -1

    frmUser.Width = Screen.Width
    frmUser.Height = Screen.Height
    frmUser.FontSize = fntButtonSize
    frmUser.FontName = fntButton
    frmUser.FontBold = fntButtonBold
    frmUser.FontItalic = fntButtonItalic
    distance = frmUser.ScaleWidth * .011

    frmUser.BackColor = colHintergrund

    SetBackgrPic

    txtAusgabe(0).FontName = fntInfoText

    'Steuer-Buttons einpassen und anordnen
    dw = 3 / 20
    dh = 1 / 15

    c&(1) = colButton
    b% = (colButton \ &H10000) And &HFF
    g% = (colButton \ &H100) And &HFF
    r% = (colButton And &HFF)
    c&(0) = RGB(r% * 1.33, g% * 1.33, b% * 1.33)
    c&(2) = RGB(r% * .67 + 1, g% * .67 + 1, b% * .67 + 1)
    c&(3) = RGB(r% * .8 + 1, g% * .8 + 1, b% * .8 + 1)

    For i% = 0 To 5
        butControl(i%).Width = frmUser.ScaleWidth * dw
        butControl(i%).Height = frmUser.ScaleHeight * dh
        butControl(i%).Left = frmUser.ScaleWidth - (2 - (i% Mod 2)) * (butControl(i%).Width + distance)
    Next i%
    fs = 0
    While (frmUser.TextWidth(butControl(kommentar).Caption) > (butControl(kommentar).Width - 8)) And (fs <> frmUser.FontSize)
        fs = frmUser.FontSize
        frmUser.FontSize = frmUser.FontSize - 1
    Wend
    butControl(kommentar).FontName = frmUser.FontName
    butControl(kommentar).FontSize = frmUser.FontSize
    For i% = 0 To 5
        butControl(i%).FontBold = frmUser.FontBold
        butControl(i%).FontItalic = frmUser.FontItalic
        If i% <> kommentar Then
            butControl(i%).FontSize = butControl(kommentar).FontSize
            butControl(i%).FontName = butControl(kommentar).FontName
        End If
    Next i%
```

```
    butControl(drucken).Top = frmUser.ScaleHeight - distance - butControl(4).Height
    butControl(uebersicht).Top = butControl(drucken).Top - distance - butControl(uebersicht).Height
    butControl(Zurueck).Top = butControl(uebersicht).Top - distance - butControl(Zurueck).Height
    butControl(mehr).Top = butControl(Zurueck).Top
    butControl(Suchen).Top = butControl(uebersicht).Top
    butControl(kommentar).Top = butControl(drucken).Top
    butControl(Suchen).Visible = (Dir$(ProgramDir + datIndexIni) <> "")

    butTheme(0).FontName = fntButton
    butTheme(0).FontSize = fntButtonSize
    butTheme(0).FontBold = fntButtonBold
    butTheme(0).FontItalic = fntButtonItalic
    butTheme(0).Width = butControl(Zurueck).Width
    butTheme(0).Height = frmUser.ScaleHeight * 2 * dh
    butTheme(0).Left = butControl(Zurueck).Left
    butTheme(0).Top = distance
    butTheme(0).BackColor = colButton
    If (r% + g% + b%) / 3 < 128 Then butTheme(0).ForeColor = &HFFFFFF
    LoadedButtons = 1

    dw = 5 / 8
    dh = .5
    frmUser.FontBold = False
    txtInfo.FontSize = frmUser.FontSize
    txtInfo.FontName = frmUser.FontName
    txtInfo.FontBold = frmUser.FontBold
    txtInfo.Left = distance
    txtInfo.Height = butTheme(0).TextHeight("Zg") + 5
    txtInfo.Top = frmUser.ScaleHeight - distance - txtInfo.Height


    i% = SetPicSize(picAusgabe, frmUser.ScaleWidth * dw, frmUser.ScaleHeight * dh)

    txtInfo.Width = frameAusgabe.Width

    butHand(rechts).Top = frameAusgabe.Top + frameAusgabe.Height + distance
    butHand(links).Top = butHand(rechts).Top
    w% = frmUser.ScaleWidth * 5 / 8
    butHand(links).Left = distance + (w% - distance) \ 2 - butHand(links).Width
    butHand(rechts).Left = butHand(links).Left + butHand(links).Width + distance

    txtAusgabe(1).Width = frameAusgabe.Width - butScroll(0).Width
    txtAusgabe(0).Width = txtAusgabe(1).Width
    butScroll(0).Left = distance + txtAusgabe(1).Width
    butScroll(1).Left = butScroll(0).Left
    Scrollbar.Width = butScroll(0).Width + 2
    Scrollbar.Left = butScroll(0).Left

    modThemePic = Val(GetINISTring("ButtonPictureMode", "3"))
    mciAVI.hWndDisplay = animAusgabe.hWnd
    ChangeDir StartDir
End Sub

Sub mciAVI_Done (NotifyCode As Integer)
    If NotifyCode = 1 Then
        If ((AnimReplay > 0) Or (AnimReplay < 0)) Then
            mciAVI.From = 0
            mciAVI.Command = "Play"
            AnimReplay = AnimReplay - 1
            If WAVWhileAnim = 2 Then
                mciWAV_Timer.Enabled = False
                mciWAV.Command = "Prev"
                mciWAV.Command = "Play"
            End If
        Else
            mciAVI.Command = "Close"
            animAusgabe.Visible = False
            If (WAVWhileAnim <> 0) And ((mciWAV.Mode = 526) Or mciWAV_Timer.Enabled) Then
                mciWAV.Command = "Close"
                SoundPlayed = True
            End If
            If Not SoundPlayed Then
                If (mciWAV.Mode = 524) Then mciWAV.Command = "Open"
                If mciWAV.Mode <> 526 Then mciWAV.Command = "Play"
            End If
            SetPic bitmap
```

```
        End If
    End If
End Sub

Sub mciWAV_Done (NotifyCode As Integer)
    If NotifyCode = 1 Then
        SoundPlayed = True
        mciWAV.From = 0
        If SoundReplay > 0 Then SoundReplay = SoundReplay - 1
        If SoundReplay = 0 Then
            mciWAV.Command = "Close"
            If picWhileWAV Then SetPic ""
        End If
        If (SoundReplay <> 0) And (SoundDelay = 0) Then mciWAV.Command = "Play"
        mciWAV_Timer.Interval = SoundDelay
        mciWAV_Timer.Enabled = (SoundReplay <> 0)
    End If
End Sub

Sub mciWAV_Timer_Timer ()
    mciWAV.Command = "Play"
    mciWAV_Timer.Enabled = False
End Sub

Sub ResetTimer_Timer ()
    minuten = minuten + 1
    Select Case minuten
    Case 1: If Thema$ <> "" Then PrintInfo strShow$ + Thema$
    Case 2: If butControl(mehr).Visible Then PrintInfo strMore$
    Case 3
        If actDecade > 0 Then
            PrintInfo strBack2$
        Else
            PrintInfo strBack1$
        End If
    Case 4: If butScroll(0).Visible Then PrintInfo strScroll$
    Case 5
        Unload frmKeyboard
        Unload frmIndex
        ReDim way(0)
        way(0).path = StartDir
        way(0).button = -1
        ChangeDir StartDir
        butControl(Zurueck).Visible = False
        ResetTimer.Enabled = False
        ResetTimer.Interval = 0
        minuten = 0
    End Select
End Sub
```

Borland Pascal 7.0 Sourcecode für MEDIA.DLL

```
library Media;
{$N+,S-,R-}

uses winprocs,wintypes,win31,OMemory;

type
    PtrRec = record
        Lo, Hi: Word
    end;
    IOFunction = function(FP: integer; Buf: PChar; Size: Integer): Word;
    TDCPalTable = Array[0..0] of TPaletteEntry;
    TBMPPalTable = Array[0..0] of TRGBQuad;
    TMetaFileHeader=record
                                Lkey,Hkey,hmf:word;
                                bbox:array[1..4] of word;
                                inch:word;
                                Lres,Hres:word;
                                check:word;
                            end;

const
    OneIO = 32768;
    BMType = $4D42;  { = 'BM' }

var
    c:array[0..5] of char;
    DCPalette:PLogPalette;
    PalSize:word;
    hPal:hPalette;



procedure AHIncr; far; external 'KERNEL' index 114;

function PCharToString(p:PChar):string;
var
    i:byte;
    s:string;

begin
    i:=1;
    while p^<>#0 do begin
        s[i]:=p^;
        inc(i);
        inc(p);
    end;
    s[0]:=chr(i-1);
    PCharToString:=s;
end;

function HugeIO(IOFunc: IOFunction; F: Integer; P: Pointer; Size: Longint): Word;
var
    L, N: Longint;

begin
    HugeIO := 1;
    L := 0;
    while L < Size do
    begin
        N := Size - L;
        if N > OneIO then N := OneIO;
        if IOFunc(F,  Ptr(PtrRec(P).Hi + PtrRec(L).Hi * Ofs(AHIncr),
                    PtrRec(L).Lo),
                    Integer(N))
            <> N then
        begin
            HugeIO := 0;
            Exit;
        end;
        Inc(L, N);
    end;
end;
```

```
function _LFileSize(F : integer) : longint;
var
   CurPos : longint;

begin
   CurPos := _llseek(F,0,1);
   _LFileSize := _llseek(F,0,2);
   _llseek(F,CurPos,0);
end;

function MedGetDCColors(dc:hDC):integer;export;
begin
   MedGetDCColors:=GetDeviceCaps(DC,BITSPIXEL);
end;


function MedGetBMPSize(filename:PChar;var width,height,BitCount:integer):integer;export;
var
   hFile: Integer;              { File-Handle für Windows File-Funktionen }
   IH: TBitmapInfoHeader;  { Windows-Bitmap-Format Infoheader }
   Header: TBitmapFileHeader;                       { Bitmap-Fileheader }

begin
   hFile := _LOpen(FileName, of_Read);              {Datei für Lesezugriff öffnen}
   if hFile = -1 then begin
      MedGetBMPSize:=1;
      Exit;       {Datei konnte nicht geöffnet werden}
   end;
   if (_LRead(hFile, @Header, SizeOf(Header)) <> SizeOf(Header))       {Bitmap Fileheader lesen}
   or (Header.bfType <> BMType) then begin
      _LClose(hFile);  {Fehler beim Lesen des Headers}
      MedGetBMPSize:=2;
      Exit;
   end;
   if (_LRead(hFile, @IH, SizeOf(IH)) <> SizeOf(IH)) then begin  {Bitmap Infoheader lesen}
      _LClose(hFile);  {Fehler beim Lesen des Headers}
      MedGetBMPSize:=2;
      Exit;
   end;
   MedGetBMPSize:=0;
   width:=IH.biWidth;
   height:=IH.biHeight;
   BitCount:=IH.biBitCount;
   _LClose(hFile);
end;

function MedProofBMP(filename:PChar):integer;export;
var
   hFile: Integer;    { File-Handle für Windows File-Funktionen }
   IH: TBitmapInfoHeader;  { Windows-Bitmap-Format Infoheader }
   Header: TBitmapFileHeader;                       { Bitmap-Fileheader }

begin
   hFile := _LOpen(FileName, of_Read);              {Datei für Lesezugriff öffnen}
   if hFile = -1 then begin
      MedProofBMP:=1;
      Exit;       {Datei konnte nicht geöffnet werden}
   end;
   if (_LRead(hFile, @Header, SizeOf(Header)) <> SizeOf(Header)) {Bitmap Fileheader lesen}
   or (Header.bfType <> BMType) then begin
      _LClose(hFile);  {Fehler beim Lesen des Headers}
      MedProofBMP:=2;
      Exit;
   end;
   if (_LRead(hFile, @IH, SizeOf(IH)) <> SizeOf(IH)) then begin {Bitmap Infoheader lesen}
      _LClose(hFile);  {Fehler beim Lesen des Headers}
      MedProofBMP:=2;
      Exit;
   end;
   MedProofBMP:=0;
   _LClose(hFile);
end;
```

```
function MedProofWAV(filename:PChar):integer;export;
type
   wave=record
               riff:array[0..3] of char;
               size:longint;
               format:array[0..6] of char;
            end;
var
   hFile: Integer;    { File-Handle für Windows File-Funktionen }
   header:wave;

begin
   hFile := _LOpen(FileName, of_Read);                 {Datei für Lesezugriff öffnen}
   if hFile = -1 then begin
      MedProofWAV:=1;
      Exit;          {Datei konnte nicht geöffnet werden}
   end;
   if (_LRead(hFile, @Header, SizeOf(Header)) <> SizeOf(Header)) then begin {RIFF lesen}
      _LClose(hFile);  {Fehler beim Lesen des Headers}
      MedProofWAV:=2;
      Exit;
   end;
   if  (Header.riff='RIFF')
   and (Header.size<>0)
   and (Header.format='WAVEfmt')
   then MedProofWAV:=0
   else MedProofWAV:= 2;
   _LClose(hFile);
end;

function MedGetAVISize(filename:PChar;var width,height:integer):integer;export;
type
   anim=record
               riff:array[0..3] of char;
               size:longint;
               format:array[0..7] of char;
            end;
   AnimSize=record
                  width,height:longint;
               end;
var
   hFile: Integer;    { File-Handle für Windows File-Funktionen }
   header:anim;
   size:AnimSize;

begin
   hFile := _LOpen(FileName, of_Read);                 {Datei für Lesezugriff öffnen}
   if hFile = -1 then begin
      MedGetAVISize:=1;
      Exit;          {Datei konnte nicht geöffnet werden}
   end;
   if (_LRead(hFile, @Header, SizeOf(Header)) <> SizeOf(Header)) then begin {RIFF lesen}
      _LClose(hFile);  {Fehler beim Lesen des Headers}
      MedGetAVISize:=2;
      Exit;
   end;
   if ((_LlSeek(hFile,$40,0))<>$40)
   or (Header.riff<>'RIFF')
   or (Header.size=0)
   or (Header.format<>'AVI LIST') then begin
      _LClose(hFile);  {Fehler beim Lesen des Headers}
      MedGetAVISize:=2;
      Exit;
   end;
   if (_LRead(hFile,@size,SizeOf(size))<>SizeOf(size)) then begin
      _LClose(hFile);  {Fehler beim Lesen des Headers}
      MedGetAVISize:=2;
      Exit;
   end;
   MedGetAVISize:=0;
   width:=size.width;
   height:=size.height;
   _LClose(hFile);
end;
```

```
function MedProofAVI(filename:PChar):integer;export;
type
  anim=record
          riff:array[0..3] of char;
          size:longint;
          format:array[0..7] of char;
        end;
var
  hFile: Integer;    { File-Handle für Windows File-Funktionen }
  header:anim;

begin
  hFile := _LOpen(FileName, of_Read);          {Datei für Lesezugriff öffnen}
  if hFile = -1 then begin
    MedProofAVI:=1;
    Exit;          {Datei konnte nicht geöffnet werden}
  end;
  if (_LRead(hFile, @Header, SizeOf(Header)) <> SizeOf(Header)) then begin {RIFF lesen}
    _LClose(hFile);  {Fehler beim Lesen des Headers}
    MedProofAVI:=2;
    Exit;
  end;
  if  (Header.riff='RIFF')
  and (Header.size<>0)
  and (Header.format='AVI LIST')
  then MedProofAVI:=0
  else MedProofAVI:= 2;
  _LClose(hFile);
end;

function MedGetWMFProp(filename:PChar;var Height_To_Width:single):integer;export;
var
  hFile: Integer;    { File-Handle für Windows File-Funktionen }
  header:TMetaFileHeader;
  key:longint;

begin
  hFile := _LOpen(FileName, of_Read);          {Datei für Lesezugriff öffnen}
  if hFile = -1 then begin
    MedGetWMFProp:=1;
    Exit;       {Datei konnte nicht geöffnet werden}
  end;
  if (_LRead(hFile, @Header, SizeOf(Header)) <> SizeOf(Header)) then begin {Header lesen}
    _LClose(hFile);  {Fehler beim Lesen des Headers}
    MedGetWMFProp:=2;
    Exit;
  end;
  with Header do begin
    key:=Hkey;
    key:=(key shl $10)+Lkey;
    if  (key=$9AC6CDD7)
    and (check=(lkey
            xor hkey
            xor hmf
            xor bbox[1]
            xor bbox[2]
            xor bbox[3]
            xor bbox[4]
            xor inch
            xor Lres
            xor Hres)) then begin
      MedGetWMFProp:=0;
      Height_To_Width:=word((bbox[4]-bbox[2])) / word((bbox[3]-bbox[1]));
    end
    else MedGetWMFProp:=2;
  end;
  _LClose(hFile);
end;
```

```
function MedProofWMF(filename:PChar):integer;export;
var
    hFile: Integer;    { File-Handle für Windows File-Funktionen }
    header:TMetaFileHeader;
    key:longint;

begin
    hFile := _LOpen(FileName, of_Read);                {Datei für Lesezugriff öffnen}
    if hFile = -1 then begin
        MedProofWMF:=1;
        Exit;          {Datei konnte nicht geöffnet werden}
    end;
    if (_LRead(hFile, @Header, SizeOf(Header)) <> SizeOf(Header)) then begin {Header lesen}
        _LClose(hFile);  {Fehler beim Lesen des Headers}
        MedProofWMF:=2;
        Exit;
    end;
    with Header do begin
        key:=Hkey;
        key:=(key shl $10)+Lkey;
        if  (key=$9AC6CDD7)
        and (check=(lkey
                xor hkey
                xor hmf
                xor bbox[1]
                xor bbox[2]
                xor bbox[3]
                xor bbox[4]
                xor inch
                xor Lres
                xor Hres))
        then MedProofWMF:=0
        else MedProofWMF:=2;
    end;
    _LClose(hFile);
end;

procedure MedShadow(DC:hDC;left,top,width,height,deep,intens:integer);export;
var
    x,y:integer;
    r,g,b:longint;
    color:TColorRef;

begin
    if intens=0 then exit;
    for x:=left+deep to left+width+deep do
        for y:=top+height to top+height+deep do begin
            color:=GetPixel(DC,x,y);
            r:=GetRValue(color);
            g:=GetGValue(color);
            b:=GetBValue(color);
{    Graustufen-Schattierungen}
{    color:=(r+g+b) DIV (3*intens);
            color:=RGB(color,color,color);
}
            color:=RGB(r div intens,g div intens,b div intens);
            SetPixel(DC,x,y,color);
        end;
    for y:=top+deep to top+height-1 do
        for x:=left+width to left+width+deep do begin
            color:=GetPixel(DC,x,y);
            r:=GetRValue(color);
            g:=GetGValue(color);
            b:=GetBValue(color);
            color:=RGB(r div intens,g div intens,b div intens);
            SetPixel(DC,x,y,color);
        end;
end;

function MedGetDC(Wnd:hWnd):hDc;export;
begin
    MedGetDC:=GetDC(Wnd);
end;
```

```
procedure CopyPalette(var DCPal:TDCPalTable; BMPPal:TBMPPalTable);
var
    i:integer;

begin
    for i:= 0 to 255 do begin
        DCPal[i].peRed   := BMPPal[i].rgbRed;
        DCPal[i].peGreen := BMPPal[i].rgbgreen;
        DCPal[i].peBlue  := BMPPal[i].rgbblue;
        DCPal[i].peFlags := PC_RESERVED;
    end; { for }
end;

procedure FillPalette(var PalTable:TDCPalTable);
var
    i:integer;
    red, green, blue:integer;

begin
    red   := 0;
    green := 0;
    blue  := 0;

    for i:= 0 to 255 do begin
        PalTable[i].peRed   := red;
        PalTable[i].peGreen := green;
        PalTable[i].peBlue  := blue;
        PalTable[i].peFlags := PC_RESERVED;

        Red := Red + 32;
        if Red > 255 then begin
            red := red mod 256;
            Green := Green + 32;
            if Green > 255 then begin
                Green:=green mod 256;
                Blue := (Blue + 64) mod 256;
            end;
        end; { if }
    end; { for }
end;


procedure MedSetBMP(DC:hDC;left,top,width,height:integer;prop:integer;filename:PChar);export;

var
    hFile: Integer;    { File-Handle für Windows File-Funktionen }
    hBmp: THandle;     { Handle für Bitmap-Speicher }
    BmpSize, ClrCnt: Longint;                    { Größe der Bitmap, Anzahl der Farben }
    pIH: PBitmapInfo;       { Windows-Bitmap-Format Infoheader }

    Header: TBitmapFileHeader;                   { Bitmap-Fileheader }
    relation:real;      {Proportionalfaktor}
    w,h,x,y:integer;
    i:byte;

begin
    hFile := _LOpen(FileName, of_Read);          {Datei für Lesezugriff öffnen}
    if hFile = -1 then Exit; {Datei konnte nicht geöffnet werden}
    if (_LRead(hFile, @Header, SizeOf(Header)) <> SizeOf(Header)) {Bitmap Fileheader lesen}
    or (Header.bfType <> BMType) then begin
        _LClose(hFile); {Fehler beim Lesen des Headers}
        Exit;
    end;
    BmpSize := _LFileSize(hFile) - SizeOf(TBitmapFileHeader); {Rest der Datei lesen}
    hBmp := GlobalAlloc(GMEM_MOVEABLE, BmpSize);         {Speicher reservieren}
    if hBmp = 0 then begin
        _LClose(hFile); {Nicht genug Speicher frei}
        Exit;
    end;
    pIH := GlobalLock(hBmp);                     {Pointer auf gesperrten Speicher setzen}
    if  (HugeIO(_LRead, hFile, pIH, BmpSize) <> 0){InfoHeader+Palette+Daten lesen}
    and (pIH^.bmiHeader.biSize = SizeOf(TBitmapInfoHeader)) then begin {gelesener Header=Var-Größe}
        ClrCnt := Header.bfOffBits - SizeOf(TBitmapFileHeader);   {Daten-Offset zu P^ berechnen}

        case prop of
        {linke obere Ecke - Originalgröße:}
```

```
    0:begin
        x:=(pIH^.bmiHeader.biWidth-width) div 2;
        y:=(pIH^.bmiHeader.biHeight-Height) div 2;
        StretchDIBits(DC,left,top,width,height,0,pIH^.bmiHeader.biHeight-Height,Width,Height,
                    Ptr(PtrRec(pIH).Hi,ClrCnt),pIH^,DIB_RGB_COLORS,SRCCOPY);
      end;
    {zentraler Ausschnitt - Originalgröße:}
    1:begin
        x:=(pIH^.bmiHeader.biWidth-width) div 2;
        y:=(pIH^.bmiHeader.biHeight-Height) div 2;
        StretchDIBits(DC,left,top,width,height,x,y,Width,Height,
                    Ptr(PtrRec(pIH).Hi,ClrCnt),pIH^,DIB_RGB_COLORS,SRCCOPY);
      end;
    {Bild genau  einpassen:}
    2:begin
        SetStretchBltMode(DC,STRETCH_DELETESCANS);
        StretchDIBits(DC,left,top,width,height,0,0,pIH^.bmiHeader.biWidth,pIH^.bmiHeader.biheight,
                    Ptr(PtrRec(pIH).Hi,ClrCnt),pIH^,DIB_RGB_COLORS,SRCCOPY);
      end;
    {Bild einpassen und Bildrelationen beibehalten :}
    3:begin
        w:=width;  {alte Werte speichern}
        h:=height;
        relation:=pIH^.bmiHeader.biWidth / pIH^.bmiHeader.biHeight;
        width:=Round(height*relation);          {Neue Breite in Relation zur Höhe setzen}
        if width>w then begin                   {Wenn neue Breite größer als gewünscht...}
          width:=w;
          height:=Round(width/relation);        {Neue Höhe in Relation zur alten Breite setzen}
        end;
        left:=left+(w-width) div 2;             {Bild zentrieren}
        top:=top+(h-height) div 2;
        SetStretchBltMode(DC,STRETCH_DELETESCANS);
        StretchDIBits(DC,left,top,width,height,0,0,pIH^.bmiHeader.biWidth,pIH^.bmiHeader.biheight,
                    Ptr(PtrRec(pIH).Hi,ClrCnt),pIH^,DIB_RGB_COLORS,SRCCOPY);  {Bildgröße anpassen}
      end;
    end;
  end;
  GlobalUnlock(hBmp); {Speicher entsichern}
  GlobalFree(hBmp);   {Speicher freigeben}
  _LClose(hFile);  {Datei schließen}
end;

procedure MedStretchImg(sourceHDC:hDC;sl,st,sw,sh:integer;destHDC:hDc;dl,dt,dw,dh:integer;wnd:hWnd);export;
var
  URgn:HRgn;
begin
  URgn:=CreateRectRgn(0,10,10,20);
  StretchBlt(destHDC,dl,dt,dw,dh,SourceHDC,sl,st,sw,sh,SRCCopy);
  RedrawWindow(Wnd,nil,URgn,RDW_UpdateNow);
end;

procedure MedCopyImg(sourceHDC:hDC;sl,st,sw,sh:integer;destHDC:hDC;dx,dy:integer);export;
begin
  BitBlt(destHDC,dx,dy,sw,sh,sourceHDC,sl,st,SRCCopy);
end;

procedure MedMoveImg(sourceHDC:hDC;sl,st,sw,sh,x,y:integer);export;
begin
  BitBlt(sourceHDC,x,y,sw,sh,sourceHDC,sl,st,SRCCopy);
end;

procedure MedScrollWnd(Wnd:hWnd;dy:integer);export;
var
  Rect,ClipRect:PRect;
begin
  Rect:=Nil;
  ClipRect:=Nil;
  ScrollWindow(Wnd,0,dy,Rect,ClipRect);
end;

procedure MedSetWMF(dc:hDC;filename:PChar);export;
var hMF:THandle;
begin
  hMF:=GetMetaFile(filename);
  PlayMetaFile(DC,hMF);
end;
```

```
exports MedGetDCColors;
exports MedProofAVI;
exports MedProofBMP;
exports MedProofWMF;
exports MedProofWAV;
exports MedGetAVISize;
exports MedGetBMPSize;
exports MedGetWMFProp;
exports MedGetDC;
exports MedSetBMP;
exports MedStretchImg;
exports MedCopyImg;
exports MedMoveImg;
exports MedScrollWnd;
exports MedShadow;
exports MedSetWMF;

begin
end.
```

Borland Pascal 7.0 Sourcecode für KIOSKDRV.DLL

```
library KIOSK;

uses Strings;

Procedure GetItemPrice(ItemNr:PChar;price:PChar);export;
begin
   StrLCopy(price,'Bitte erfragen',14);
end;

procedure GetItemInvent(ItemNr:PChar;invent:PChar);export;
begin
   StrLCopy(invent,'Bitte erfragen',14);
end;

exports GetItemPrice;
exports GetItemInvent;

begin
end.
```